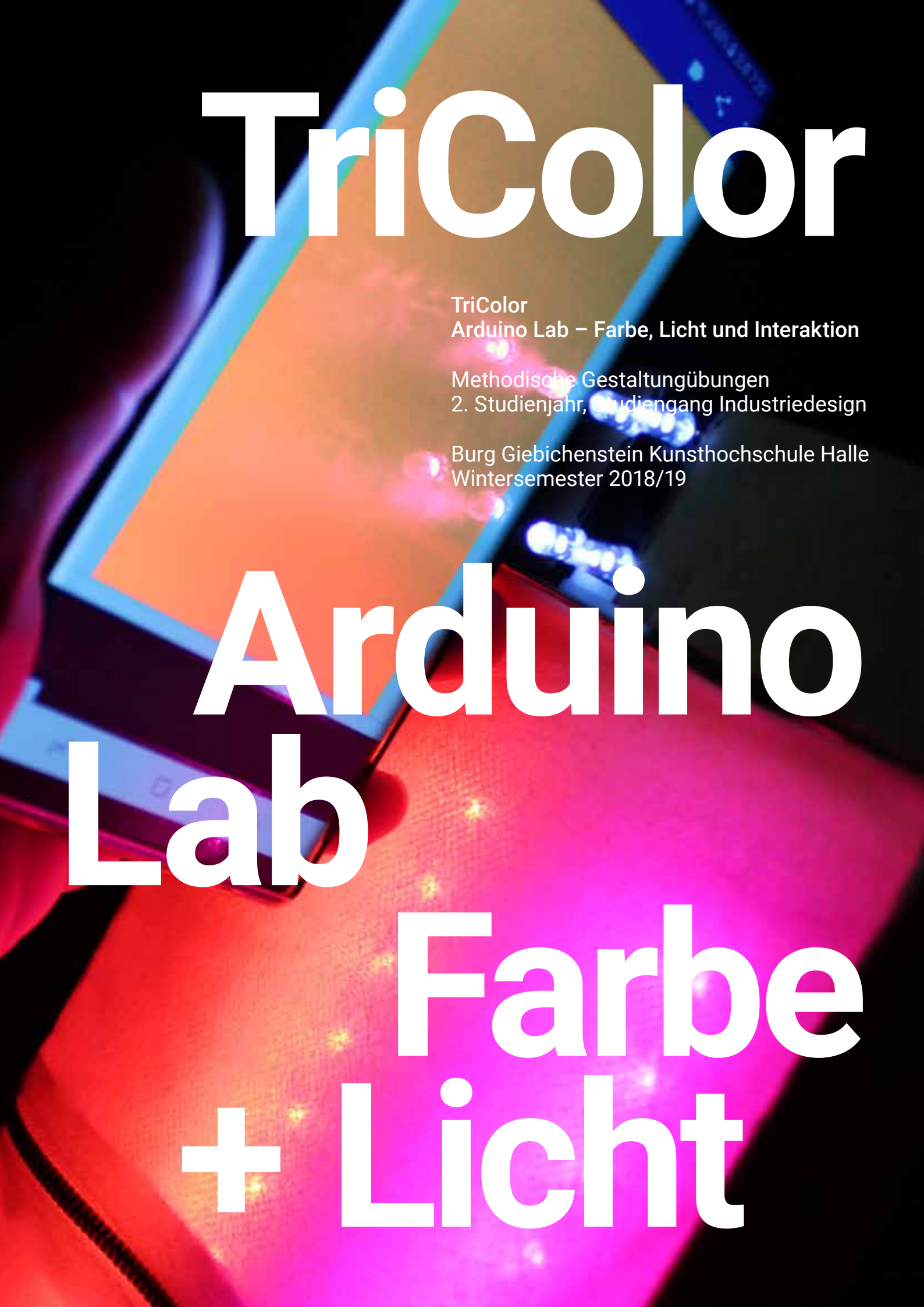


TriColor
Arduino Lab – Farbe, Licht und Interaktion

Methodische Gestaltungübungen
2. Studienjahr, Studiengang Industriedesign

Burg Giebichenstein Kunsthochschule Halle
Wintersemester 2018/19

TriColor

Arduino Lab

Farbe + Licht

Arduino Workshop

Farbe, Licht und Interaktion

Max von Elverfeldt wird in diesem Workshop in die Steuerung
von Hardware durch Arduino einführen …

Steuerung von Lichtfarbe und Lichtstärke, die Interaktion von Licht mit …
Bewegung, Geräuschen … und weiteren sensorischen Ereignissen …
dies sind nur einige der Elemente, die in diesem Workshop untersucht,
probiert und zu funktionierenden Installationen geführt werden.

TriColor, farbstark –
Experimente und Entwürfe zu Farbe als
Gestaltungsmittel in drei Sessions

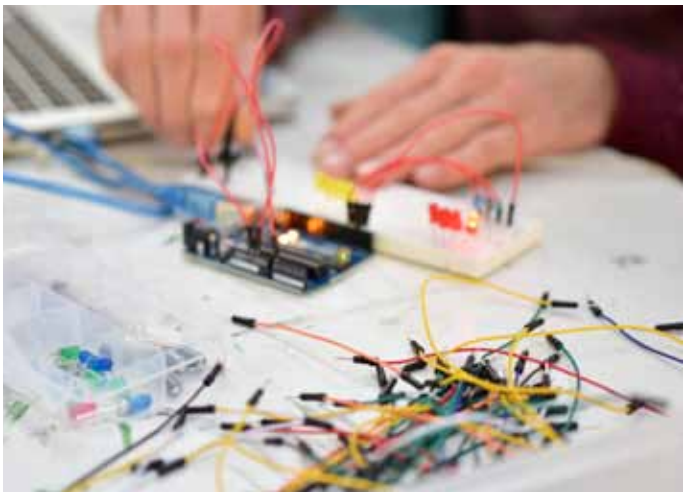
Methodische Gestaltungsübungen
Studiengang Industriedesign

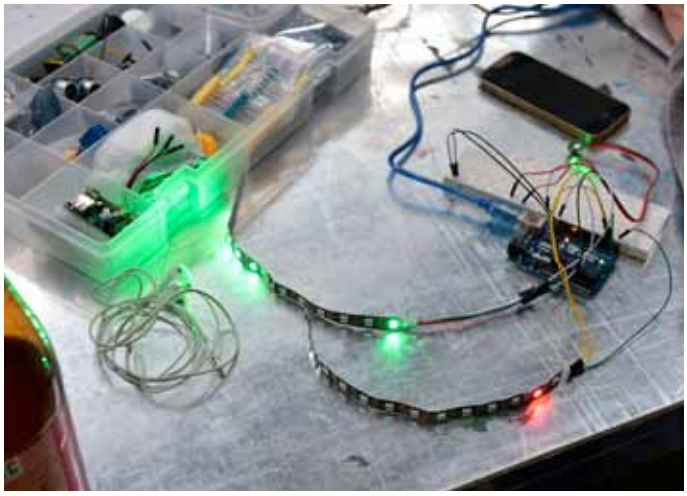
Teilnehmer_innen
Wayra Aguilar, Milan Behrens, Leonhard Burmester, Anna Freudenberg,
Dongyoung Hwang, Alina-Sophie Karre, Pierre Lichtenstein, Viola Nauck,
Nikolaus Hößle, Fridolin Richter, Anniek Timmermann

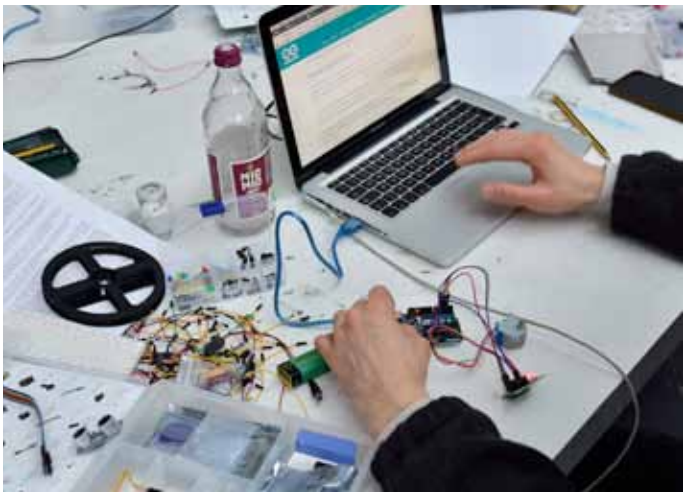
Moderation
Prof. Guido Englich, MA Benjamin Schief

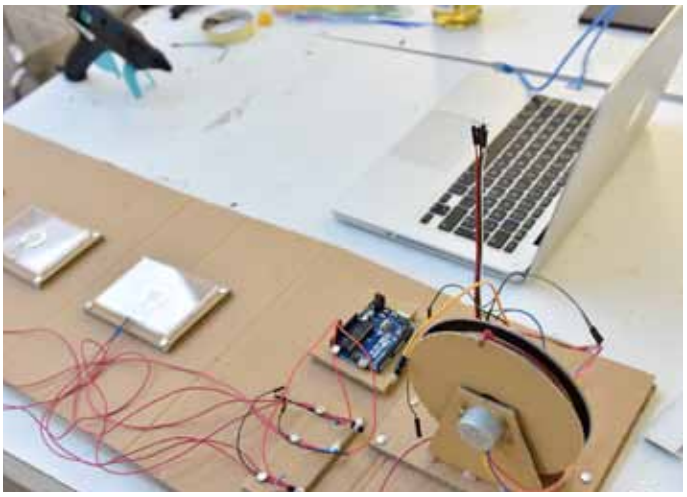
Burg Giebichenstein Kunsthochschule Halle
Wintersemester 2018/19

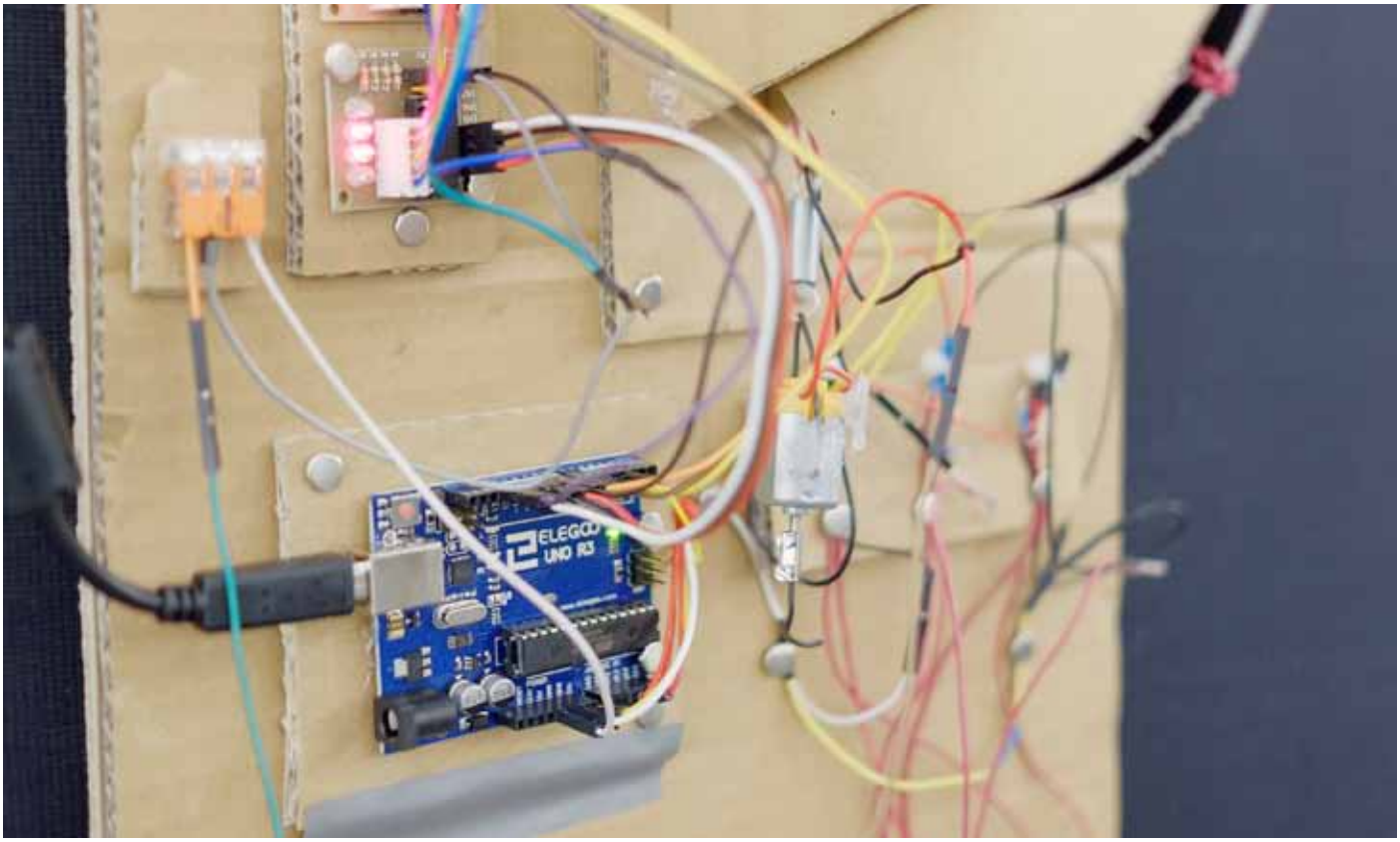
Arbeitsprozess











Entwürfe

Der launische Weihnachtsbaum

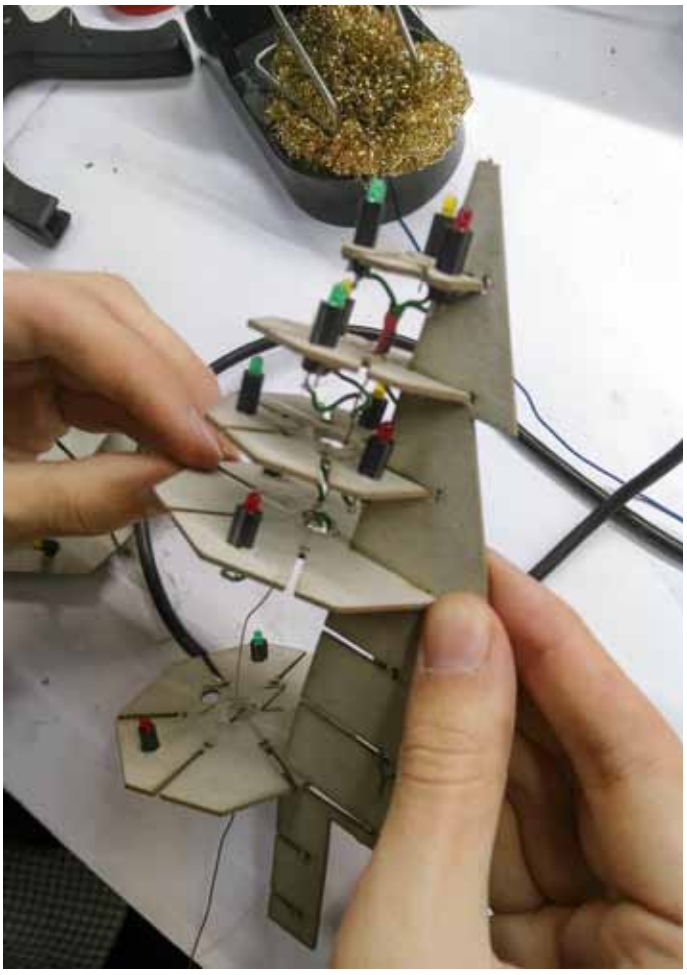
Projekt von Anna Freudenberg

Programmcode

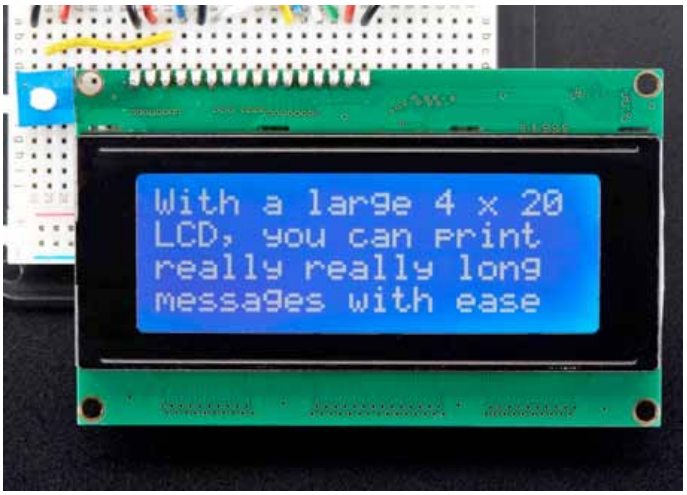
```
#include <LiquidCrystal.h>

int Geschenk = A0;
int LEDrot = 8;
int LEDgelb = 10;
int LEDgruen = 9;
bool froheWeihnachtText = false;

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

 // set up the LCD's number of columns and rows:
  lcd.begin(20, 4);
  // put your setup code here, to run once:
Serial.begin(9600);

pinMode (LEDrot, OUTPUT);
pinMode (LEDgelb, OUTPUT);
pinMode (LEDgruen, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

Serial.println(analogRead(Geschenk));

 lcd.setCursor(1, 0);
     lcd.println(„Frohe Weihnachten“);
     lcd.setCursor(0, 4);
     lcd.print(„Liebe Tricolori“);

if (analogRead(Geschenk) > 1000) {
    // turn LED on:
    digitalWrite(LEDrot, 0);
    digitalWrite(LEDgelb, 0);
    digitalWrite(LEDgruen, 0);

    froheWeihnachtText=true;
  } else {
    // turn LED off:
    froheWeihnachtText=false;
    digitalWrite(LEDrot, 1);
    digitalWrite(LEDgelb, 1);
    digitalWrite(LEDgruen, 1);
    Serial.println(analogRead(Geschenk));

  }

 if(true) {
    }

    }
```

Der launische Weihnachtsbaum

Projekt von Anna Freudenberg

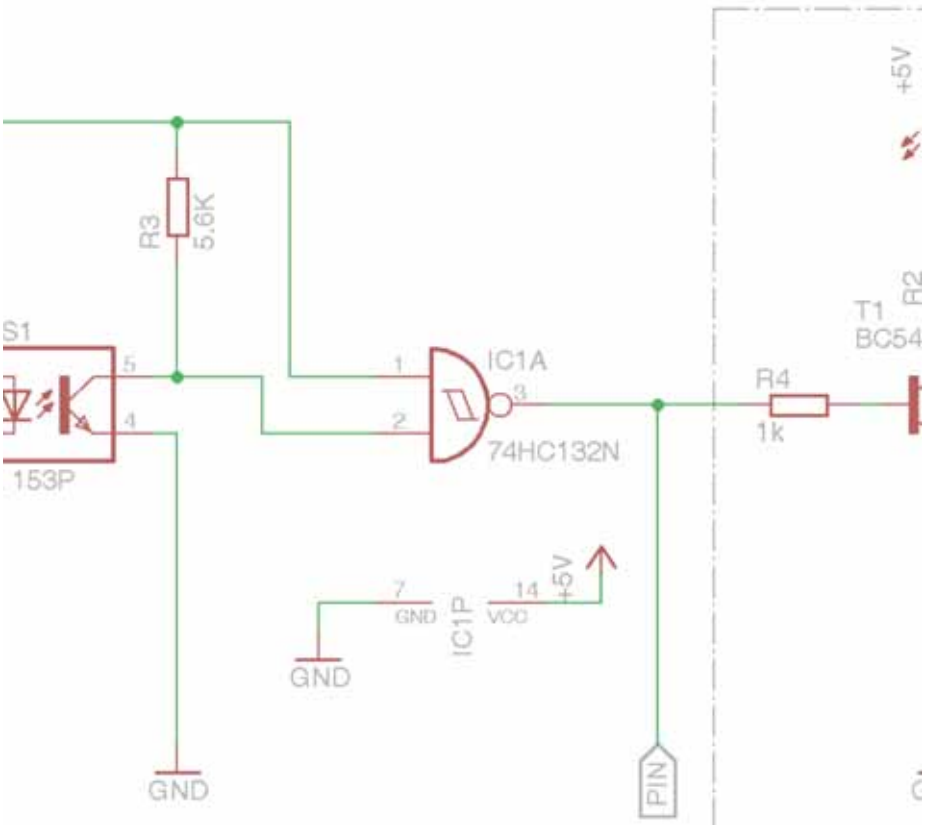


Ausgehend von der Ampelschaltung im Stadtverkehr wurde die Idee entwickelt ein „Farbschaltprogramm“ für den Weihnachtsbaum zu entwickeln. Der Weihnachtbaum hat eine Lichterkette mit verschiedenen Farben (rot, grün, gelb), welche jedoch nur unter bestimmten Bedingungen leuchten. Sind diese nicht erfüllt oder werden diese verändert kommt es zu keiner Farbdarstellung.

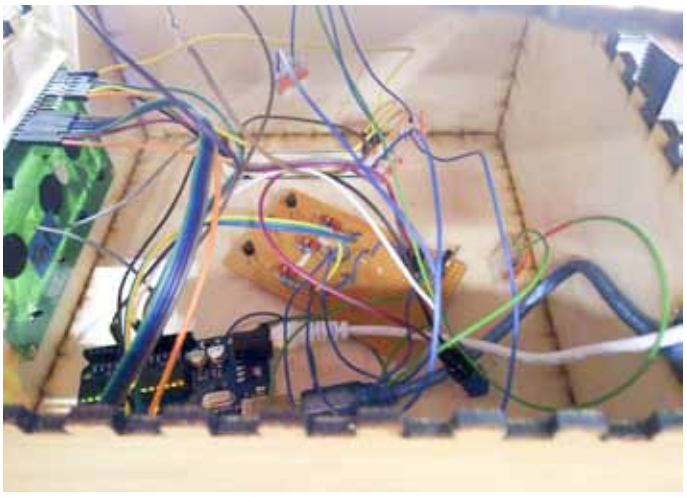


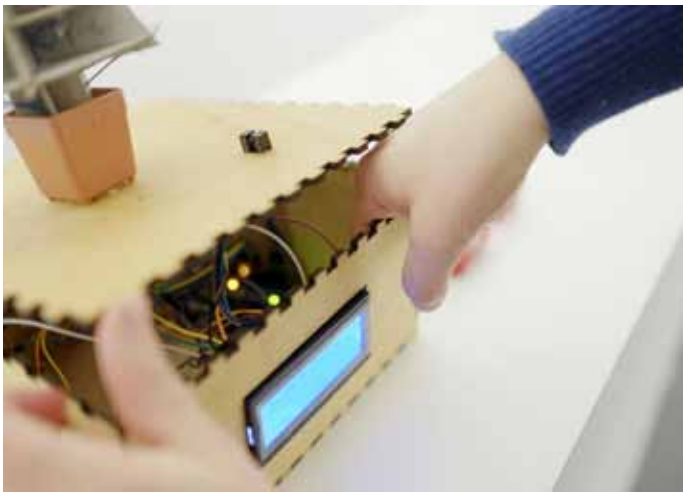
Der Weihnachtsbaum muss zunächst in Stimmung gebracht werden: Im Weihnachtskontext sind vielzählige Möglichkeiten denkbar, welche man jeweils in verschiedene Sensorinteraktionen mit den Baumlichtern übersetzen könnte (Z.B.: Kerzen-Wärme-Temerpatursensor). Nach einer Phase des Austestens entschied ich mich für den Einsatz einer Gabellichtschranke, welche das Leuchten der Baumlichter auslöst, sobald das „passende" Geschenk unter den Baum gelegt wird.

Außerdem soll anschließend eine LCD-anzeige aufleuchten, auf welchem die Aufschrift „fröhliche Weihnachten" zu lesen ist.



Der Aufbau einer Lichtschranke ist nicht sehr schwer. Vereinfacht ausgedrückt haben wir nur zwei Zustände: Der Infrarotstrahl ist nicht unterbrochen (=0), oder der Strahl wurde unterbrochen (=1). Liegt ein Geschrenk in der Schranke, kann Strom fließen (=1) und es leuchtet.

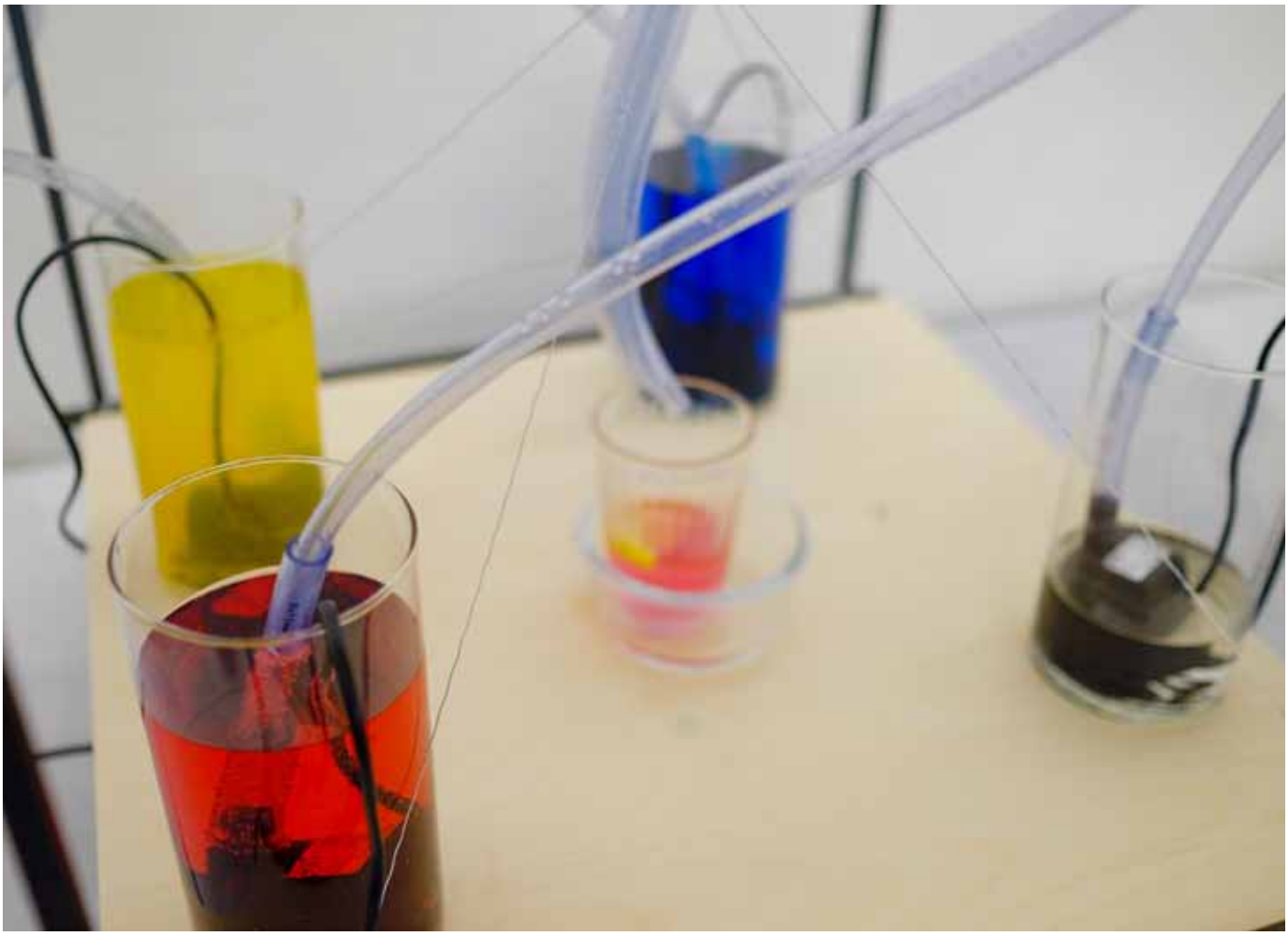




Nachdem der Tannenbaum aus Pappe gelasert und zusammengebaut wurde, ging es an die aufwendige Lötung aller 24 LED-lichter (8 jeder Farbe, rot, gelb und grün). Die kleinen LED-lichter sind alle an den Pluspol verbunden und haben jeweils eine einzelne Masse bzw. Minuspol.

Anschließend werden die Kabel durch das angefertigte Bohrloch in der Mitte des Weihnachtsbaum heruntergeführt. Dann werden die einzelnen Kabel auf dem Breadboard zusammengesteckt und mit Kleber fixiert.

Nach der Inbetriebnahme des Displays, wird die gesamte Verkabelung in einer Holzbox kaschiert.



Freudscher Icebreaker

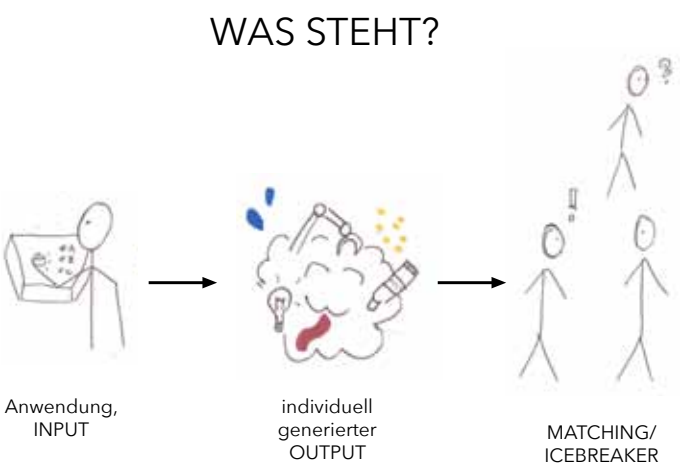
Experiment von Anniek Timmermann

Programmcode

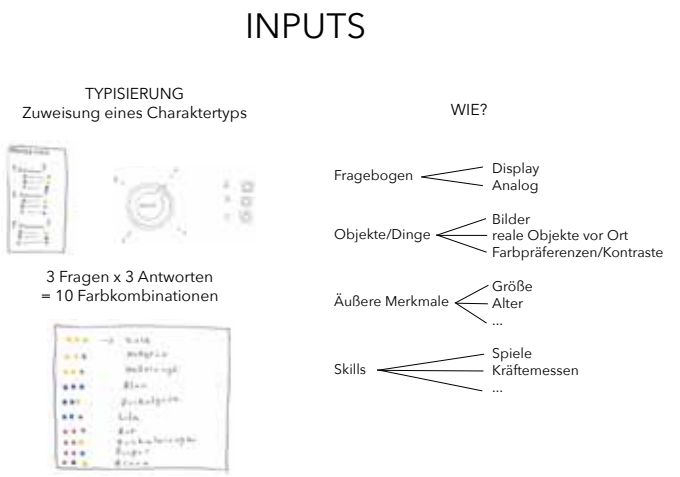
```
int Dreh1 = A0; //Poti
int Dreh1P = 0; // Variable Poti Wert

int TastA = 7; //Taster A
int TastB = 4; //Taster B
int TastC = 2; //Taster C
int TastAP = 0;
int TastBP = 0;
int TastCP = 0;

int PumpR = 11; //Pumpe Rot
int PumpG = 10; //Pumpe Gelb
int PumpB = 9;  //Pumpe Blau
int PumpT = 6;  //Pumpe Transparent
int interval = 1000;

void setup() {
  Serial.begin(9600);

  pinMode(Dreh1, INPUT);

  pinMode(TastA, INPUT);
  pinMode(TastB, INPUT);
  pinMode(TastC, INPUT);
  pinMode(PumpR, OUTPUT);
  pinMode(PumpG, OUTPUT);
  pinMode(PumpB, OUTPUT);
  pinMode(PumpT, OUTPUT);
  digitalWrite(PumpR, HIGH);
  digitalWrite(PumpG, HIGH);
  digitalWrite(PumpB, HIGH);
  digitalWrite(PumpT, HIGH);

}
void loop() {
  Dreh1P = analogRead(Dreh1); //Potiwert auslesen

  //Sektor Bereich Frage 1
  if((Dreh1P > 1) && (Dreh1P<150)){
     Serial.println(„Drehregler im ersten Bereich)“);
     TastAP = digitalRead(TastA);
     TastBP = digitalRead(TastB);
     TastCP = digitalRead(TastC);
     //Taster A
     if(TastAP == LOW){
       Serial.println(„Taster A gedrückt.)“);
       digitalWrite(PumpR, LOW);
       delay(interval*4);
       digitalWrite(PumpR, HIGH);
     }
     //Taster B
     if(TastBP == LOW){
       Serial.println(„Taster B gedrückt.)“);
       digitalWrite(PumpG, LOW);
       delay(interval*2);
       digitalWrite(PumpG, HIGH);
     }
     //Taster C
     if(TastCP == LOW){
       Serial.println(„Taster C gedrückt.)“);
       digitalWrite(PumpB, LOW);
       delay(interval*3.5);
       digitalWrite(PumpB, HIGH);
     }
    }
  //Sektor Bereich Frage 2
  if((Dreh1P > 151) && (Dreh1P<512)){
     Serial.println(„Drehregler im zweiten Bereich)“);
     TastAP = digitalRead(TastA);

     TastBP = digitalRead(TastB);
     TastCP = digitalRead(TastC);
     //Taster A
     if(TastAP == LOW){
       Serial.println(„Taster A gedrückt.)“);
       digitalWrite(PumpB, LOW);
       delay(interval*3.5);
       digitalWrite(PumpB, HIGH);
     }
     //Taster B
     if(TastBP == LOW){
       Serial.println(„Taster B gedrückt.)“);
       digitalWrite(PumpG, LOW);
       delay(interval*2);
       digitalWrite(PumpG, HIGH);
     }
     //Taster C
     if(TastCP == LOW){
       Serial.println(„Taster C gedrückt.)“);
       digitalWrite(PumpR, LOW);
       delay(interval*4);
       digitalWrite(PumpR, HIGH);
     }
   }
  //Sektor Bereich Frage 3
  if((Dreh1P > 513) && (Dreh1P<850)){
     Serial.println(„Drehregler im dritten Bereich)“);
     TastAP = digitalRead(TastA);
     TastBP = digitalRead(TastB);
     TastCP = digitalRead(TastC);
     //Taster A
     if(TastAP == LOW){
       Serial.println(„Taster A gedrückt.)“);
       digitalWrite(PumpG, LOW);
       delay(interval*2);
       digitalWrite(PumpG, HIGH);
     }
     //Taster B
     if(TastBP == LOW){
       Serial.println(„Taster B gedrückt.)“);
       digitalWrite(PumpR, LOW);
       delay(interval*4);
       digitalWrite(PumpR, HIGH);
     }
     //Taster C
     if(TastCP == LOW){
       Serial.println(„Taster C gedrückt.)“);
       digitalWrite(PumpB, LOW);
       delay(interval*3.5);
       digitalWrite(PumpB, HIGH);
     }
   }
  //Sektor Bereich Frage 4
  if((Dreh1P > 851) && (Dreh1P<1023)){
     Serial.println(„Drehregler im vierten Bereich)“);
     TastAP = digitalRead(TastA);
     TastBP = digitalRead(TastB);
     //Taster A
     if(TastAP == LOW){
       Serial.println(„Taster A gedrückt.)“);
       digitalWrite(PumpT, LOW);
       delay(interval*0.7);
       digitalWrite(PumpT, HIGH);
     }
     //Taster B
     if(TastBP == LOW){
       Serial.println(„Taster B gedrückt.)“);
       }
   }
}
```

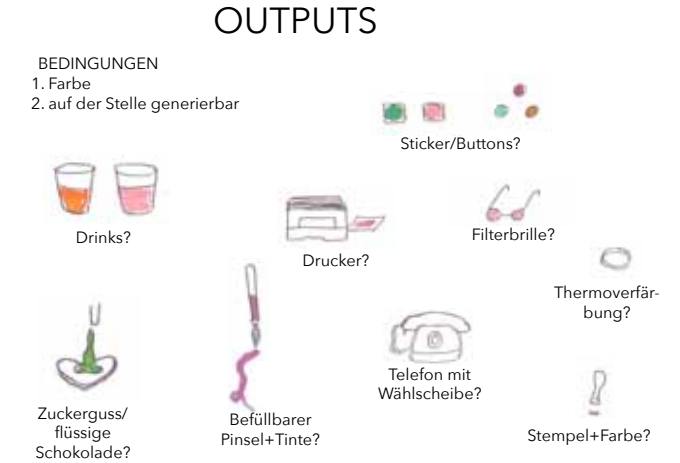
Freudscher Icebreaker

Experiment von Anniek Timmermann



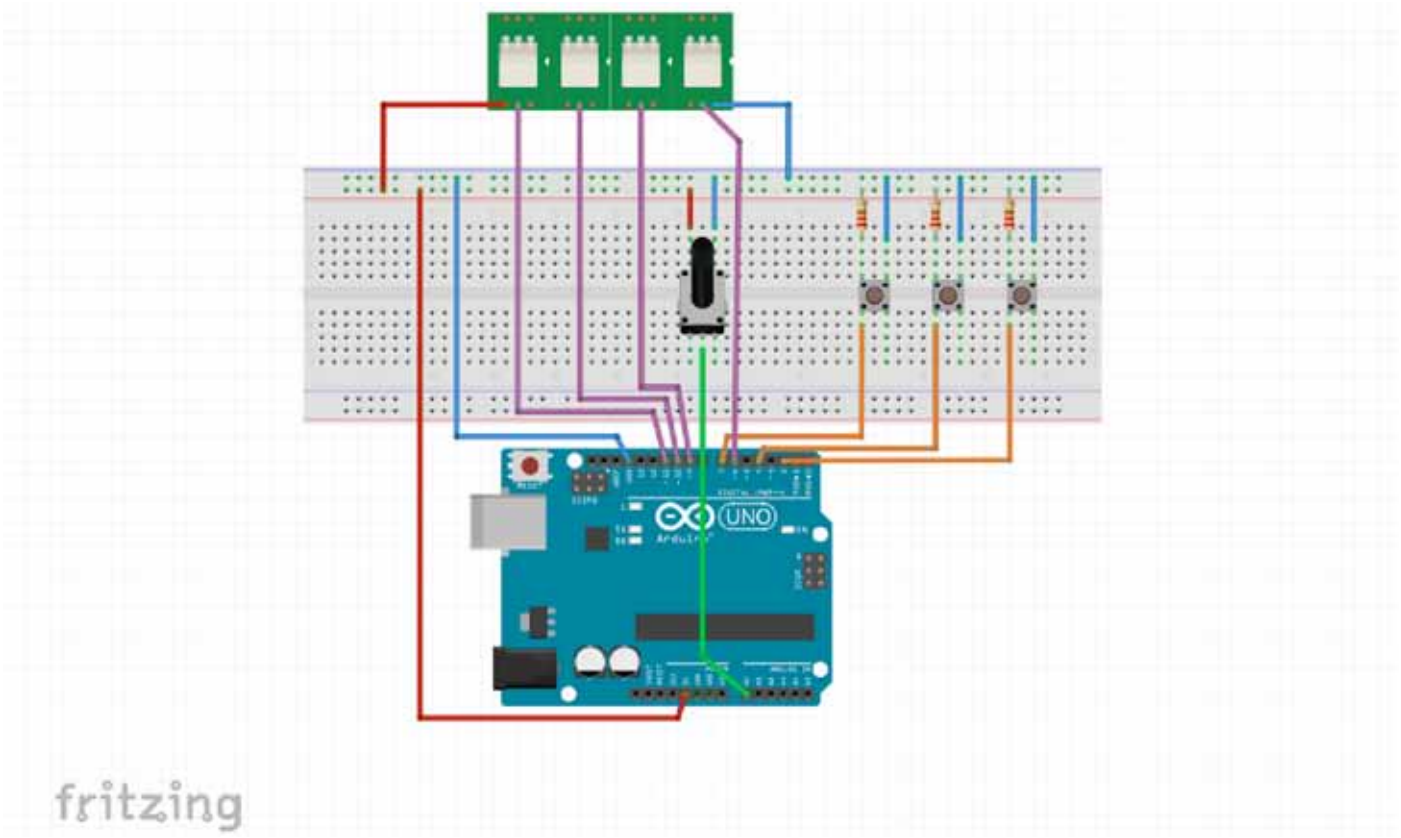
Ideenfindung während des Workshops

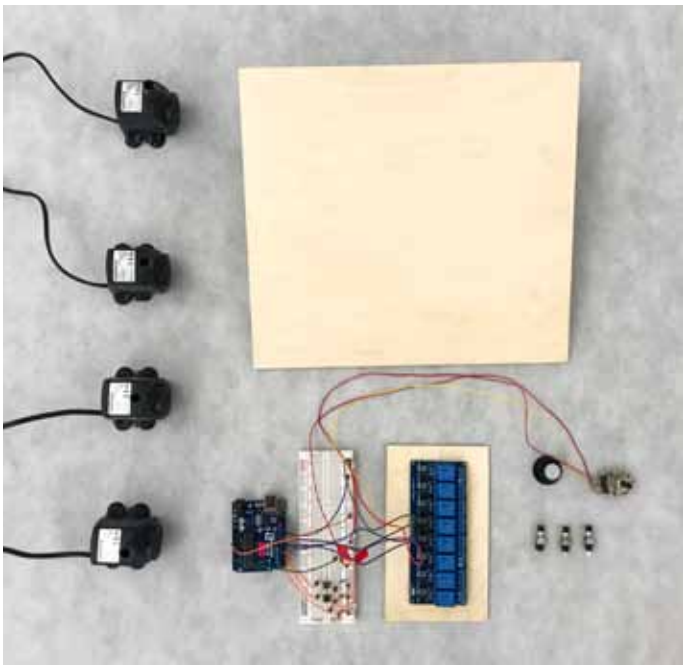




Mithilfe meines elektronischen Prototyps wollte ich Farbe als ein Merkmal einsetzen, welches den Ausgangspunkt für einen Gesprächsaufhänger darstellt. Durch die Anwendung sollte jedem Nutzer eine individuell generierte Farbe zugewiesen werden. Auf welcher Grundlage die Daten der „Typisierung“ erhoben werden, war relativ schnell beschlossen. Ein psychoanalytischer Fragebogen (in diesem Fall pseudo-wissenschaftlich), sollte zur Generierung der eigenen Farbe dienen. Die verschiedenen Antwortmöglichkeiten wurden zu diesen Zweck mit unterschiedlichen Farben – Rot, Gelb oder Blau – im Code hinterlegt.

Breadboard – Violett: Pumpen; Grün: Potentiometer (Drehregler); Orange: Taster A, B, C



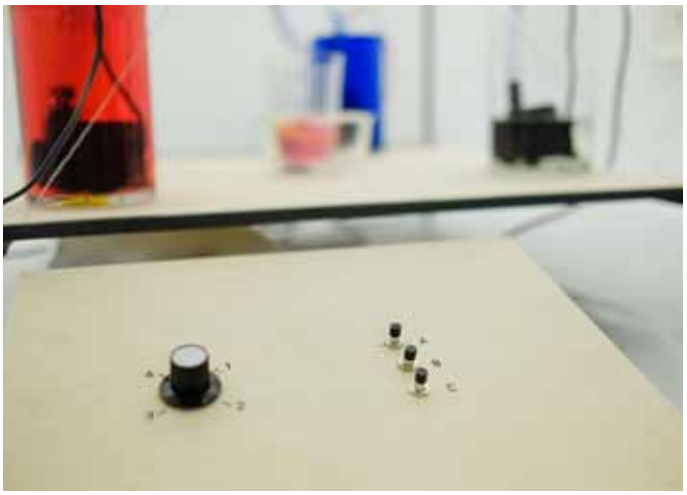


Hardware

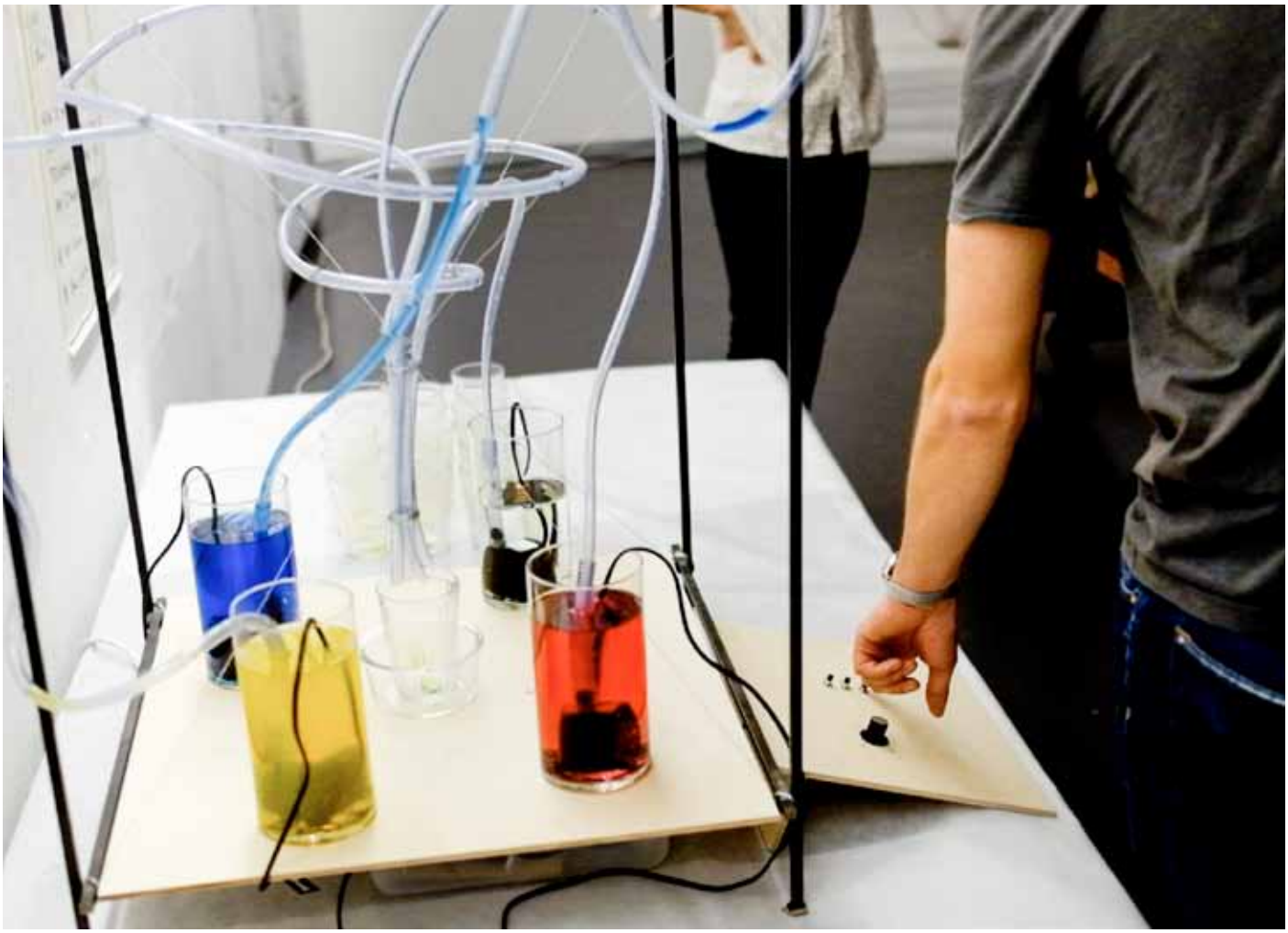
Nachdem dieser Grundgedanke feststand, ging es darum, einen geeigneten Output zu finden, beziehungsweise eine Möglichkeit, wie sich die Typisierung darstellen ließ. Dabei musste die Bedingung erfüllt werden, dass ein farbliches Ergebnis sofort nach dem Beantworten der Fragen erzeugt wird. Letztendlich haben sich dafür Flüssigkeiten angeboten, weil sich diese sofort vermischen. In Form eines Drinks ließ sich in einem sozialen Kontext die eigene Farbe mit herumtragen als auch die der anderen sichten.

Der finale Prototyp besteht aus einer Apparatur, deren Interface beziehungsweise Mikrocontroller nach Beantwortung einer Frage eine von vier Pumpen ansteuert.


5. WELCHE DIESER DREI FARBEN
GETRÄNKE DIESEN AM WENIGSTEN
A Gelb
B Rot
C Blau
4. WENN SIE AN EINER GESELL-
SCHAFTLICHEN VERANSTALTUNG
TEILNEHMEN TRINKEN SIE
IHR GETRÄNK GEWÖHNLICH
A Mit Schuss
B Ohne Schuss



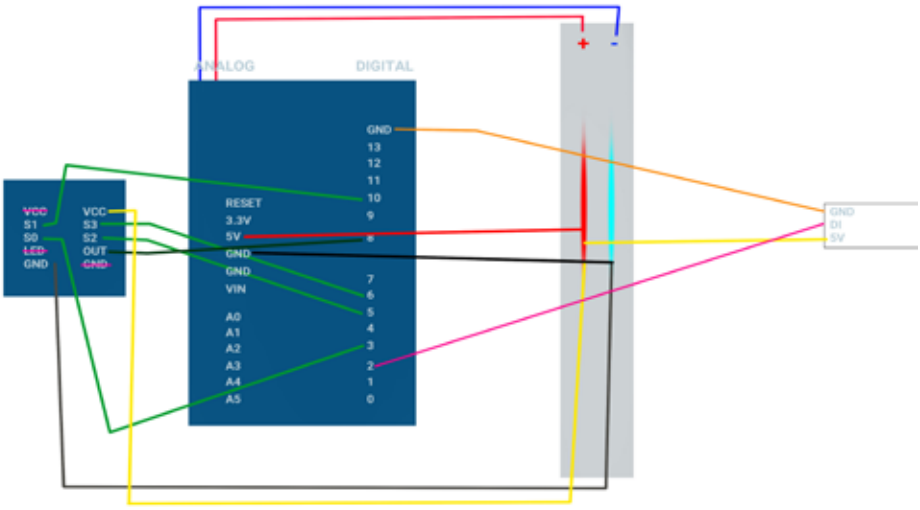
Diese befördern eine bestimmte Menge an eingefärbter Flüssigkeit durch ein Schlauchsystem und sammeln sich schließlich in einem Glas, wo sich die Farben vermischen. Der Drehregler ist mit Werten hinterlegt, welche mit dem Fragebogen an der Wand abgestimmt sind. Die Fragen wiederum sind mit Antwortmöglichkeiten A, B oder C zu beantworten. Insgesamt können so bis zu zehn unterschiedliche Drinks erstellt werden. Tonic Water wurde mit Lebensmittelfarbe eingefärbt, bei der transparenten Flüssigkeit handelt es sich um Gin, diese Flüssigkeit hat also neben dem geschmacklichen nur Unterhaltungswert und trägt nicht zur Farberstellung bei.



```
    //Taster B
    if(TastBP == LOW){
      Serial.println(„Taster B gedrückt.“);
      digitalWrite(PumpG, LOW);
      delay(interval*2);
      digitalWrite(PumpG, HIGH);
    }
    //Taster C
    if(TastCP == LOW){
      Serial.println(„Taster C gedrückt.“);
      digitalWrite(PumpB, LOW);
      delay(interval*3.5);
      digitalWrite(PumpB, HIGH);
    }
  }
 //Sektor Bereich Frage 2
 if((Dreh1P > 151) && (Dreh1P<512)){
    Serial.println(„Drehregler im zweiten Bereich)“);
    TastAP = digitalRead(TastA);

    TastBP = digitalRead(TastB);
    TastCP = digitalRead(TastC);
    //Taster A
    if(TastAP == LOW){
      Serial.println(„Taster A gedrückt.)“);
      digitalWrite(PumpB, LOW);
      delay(interval*3.5);
      digitalWrite(PumpB, HIGH);
    }
    //Taster B
    if(TastBP == LOW){
      Serial.println(„Taster B gedrückt.)“);
      digitalWrite(PumpG, LOW);
      delay(interval*2);
      digitalWrite(PumpG, HIGH);
    }
    //Taster C
    if(TastCP == LOW){
      Serial.println(„Taster C gedrückt.)“);
      digitalWrite(PumpR, LOW);
      delay(interval*4);
      digitalWrite(PumpR, HIGH);
    }
  }
 //Sektor Bereich Frage 3
 if((Dreh1P > 513) && (Dreh1P<850)){
    Serial.println(„Drehregler im dritten Bereich)“);
    TastAP = digitalRead(TastA);
    TastBP = digitalRead(TastB);
    TastCP = digitalRead(TastC);
    //Taster A
    if(TastAP == LOW){
      Serial.println(„Taster A gedrückt.)“);
      digitalWrite(PumpG, LOW);
      delay(interval*2);
      digitalWrite(PumpG, HIGH);
    }
    //Taster B
    if(TastBP == LOW){
      Serial.println(„Taster B gedrückt.)“);
      digitalWrite(PumpR, LOW);
      delay(interval*4);
      digitalWrite(PumpR, HIGH);
    }
    //Taster C
    if(TastCP == LOW){
      Serial.println(„Taster C gedrückt.)“);
      digitalWrite(PumpB, LOW);
      delay(interval*3.5);
      digitalWrite(PumpB, HIGH);
    }
  }
 //Sektor Bereich Frage 4
 if((Dreh1P > 851) && (Dreh1P<1023)){
    Serial.println(„Drehregler im vierten Bereich)“);
    TastAP = digitalRead(TastA);
    TastBP = digitalRead(TastB);
    //Taster A
    if(TastAP == LOW){
      Serial.println(„Taster A gedrückt.)“);
      digitalWrite(PumpT, LOW);
      delay(interval*0.7);
      digitalWrite(PumpT, HIGH);
    }
    //Taster B
    if(TastBP == LOW){
      Serial.println(„Taster B gedrückt.)“);
      }
   }
}
```

Camäl-Neon

Experiment von Dongyoung Hwang



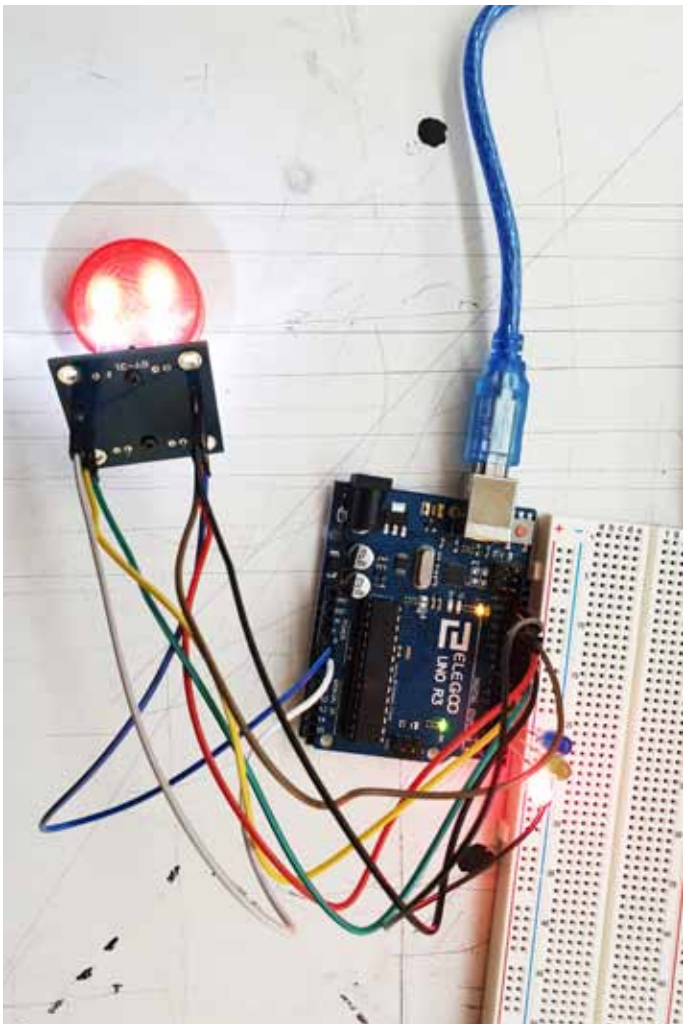
Programmcode

```
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>
#define PIN 2  //LED
Adafruit_NeoPixel strip = Adafruit_NeoPixel(49, PIN,
NEO_GRB + NEO_KHZ800);  //
const int outputEnabled = 2; // write LOW to turn on
Note, may not be hooked up.
const int s0 = 3;
const int s1 = 10;
const int s2 = 5;
const int s3 = 6;
const int out = 8;
// LED pins connected to Arduino
// Variables
int red = 0;
int green = 0;
int blue = 0;
//
int reda=0;
int greena = 0;
int bluea = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(outputEnabled, OUTPUT);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(s2, OUTPUT);
  pinMode(s3, OUTPUT);
  pinMode(out, INPUT);
   digitalWrite(s0, HIGH);
  digitalWrite(s1, HIGH);
  digitalWrite(outputEnabled, LOW);
  #if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000)
clock_prescale_set(clock_div_1);
  #endif
  // End of trinket special code
  strip.begin();
  strip.show(); // Initialize all pixels to ‚off'
}
  void loop()
{
  color();

reda = 255-red;
 bluea = 255-blue;
 greena = 255-green;
 //Serial.println();
  if (reda > greena && greena > bluea)
 {
  Serial.println(„RG“);
  reda = 255;
  greena = greena / 2;
  bluea =0;
   }
 if (reda > bluea && bluea > greena)
 {   Serial.println(„RB“);
    reda = 255;
  bluea = bluea /2;
  greena =0;
   }
 if (greena > reda && reda > bluea)
 {   Serial.println(„GR“);
    greena = 255;
  reda = reda/2;
  bluea =0;
    }if (greena > bluea && bluea > reda)
 {   Serial.println(„GB“);

  greena = 255;
  bluea = bluea/2;
  reda =0;
    }if (bluea > greena && greena > reda)
 {   Serial.println(„BG“);
    bluea = 255;
  greena = greena/2;
  reda =0;
    }if (bluea > reda && reda > greena)
 {   Serial.println(„BR“);
    bluea = 255;
  reda = reda/2;
  greena =0;
    }
 else{
 Serial.println();
 }
 Serial.print(„R Intensity:“);
 Serial.print(reda);
 Serial.print(„ G Intensity: „);
 Serial.print(greena);
 Serial.print(„ B Intensity : „);

 Serial.println(bluea);
  delay(300);
colorWipe(strip.Color(reda, greena, bluea), 50);
}

 void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
     strip.setPixelColor(i, c);
     strip.show();
     delay(wait);
  }
}
    uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
   return strip.Color(255 - WheelPos * 3, 0, WheelPos
* 3);
  } else if(WheelPos < 170) {
    WheelPos -= 85;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos
* 3);
  } else {
     WheelPos -= 170;
   return strip.Color(WheelPos * 3, 255 - WheelPos *
3, 0);
  }
}
void color()
{
  digitalWrite(s2, LOW);
  digitalWrite(s3, LOW);
  //count OUT, pRed, RED
  red = pulseIn(out, LOW);
 red /400-1;
  digitalWrite(s3, HIGH);
  //count OUT, pBLUE, BLUE
  blue = pulseIn(out, LOW);
  blue /400-1;
  digitalWrite(s2, HIGH);
  //count OUT, pGreen, GREEN
  green = pulseIn(out, LOW);
  green /400-1;
}
```

Camäl-Neon

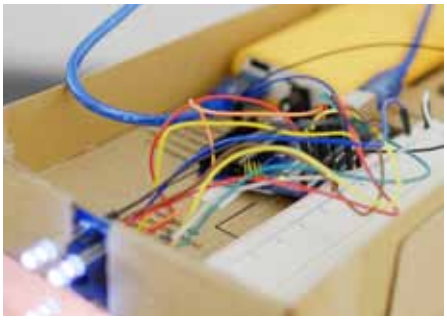
Experiment von Dongyoung Hwa

Ausgangspunkt ist die Farbeveränderung des menschlichen Gesichts. Die Idee ist, dass der Helm durch Farbesonsor die Farben erkennen und kopieren . Die durch Farbesensor erkannten Farben werden in LED beleuchtet . Man kann Farben in der Umgebung kopeiren und seine Gesichtfarben verändern wie Camäleon.



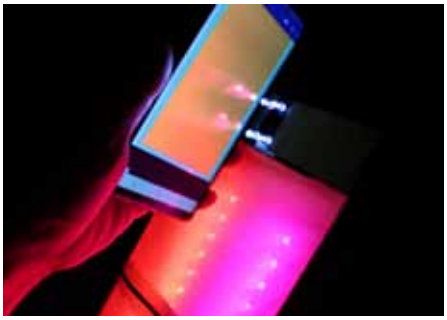
Nach vielen weiteren Übungen geht es um das Planen und Umsetzen meinen eigenen „Gesichts“. Ein Sensor soll zwischen einem Input (Farbe) und einem Output (Beleuchtung) vermitteln. Nach intesiven Arduinoausprobephasen und einigen schnellen Methoden komme ich zu einer Prototypesphase. Ich habe versucht, die Farbeveränderung als unser Gesicht im Alltag zu sehen. Und wie könnten die Anderen meine Gesichtfarbeänderung interpretieren.

https://www.instagram.com/dongyoungh/?hl=ko



Ich befestige einen Colorsensor vor der Stirn der Aussenseite des Helms und platziere die Steckboard auf dem Scheitel wie ein Gehirn. Dazu kommen Power Bank und viele Kabel. Damit man Vorne sehen kann, habe ich einen Reissverschluss aufgeklebt. Wenn der Reissverschluss offen ist, können die Anderen meine LED-Augen sehen.

Dann ab jetzt können Sie Ihre eigene Farbe mir zeigen und vor meiner Stirn näher anlegen. Ich kopiere Ihre Farbe!



Man kann die farbigen Dingen vor dem Sensor nähern. Nach der Farbeerkennung wird mein Gesichtfarbe verändert. Z.b. Wenn man eine rote Farbe vor meiner Stirn näher angelegt? Ya.Dann wird mein Gesicht Rot leuchten . Wenn man mir Blau gezeigt hat, dann wird LED Blau beleuchten. Wenn Sie mir Ihre Farbe geben, teile ich Ihre Farbe mit

Der Colorsensor funktioniert in verschiedenen Farben auch gut. Es wäre effektiv, die Farbe des Bildschirms oder Mobiltelefons anzuzeigen, damit Colorsensor den Farben richtig gut reagieren kann.

Aura Couleur

Experiment von Leonhard Burmester und Fridolin Richter

Programmcode

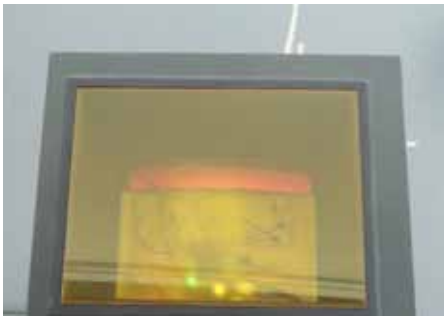
```
#include <Adafruit_NeoPixel.h>

#define N_LEDS 121
#define PIN     4

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_
LEDS, PIN, NEO_GRB + NEO_KHZ800);

int Spot = 2;
int Nebel = 7;
int Rot = 9;
int Gruen = 10;
int Blau = 11;
int led = 4;

int light = A0; //Pin Nummer für Input-Pin. Diesmal
Analog,
int light1 = A1;
int light2 = A2;
int a = 0; //Variable, um den Wert zu speichern.
int b = 0;
int c = 0;
int _from = 700;
int _to = 1024;
int randy = 0;

void setup() {

  pinMode(Rot, OUTPUT);
  pinMode(Gruen, OUTPUT);
  pinMode(Blau, OUTPUT);
  pinMode(Spot, OUTPUT);
  pinMode(Nebel, OUTPUT);
  pinMode(led, OUTPUT);

  pinMode(light, INPUT); //Definieren des InPins als
lesenden Pin.
  pinMode(light1, INPUT);
  pinMode(light2, INPUT);


  Serial.begin(9600); //Starten des seriellen
Monitors.
  strip.begin();

}

void loop() {

  Serial.print(„Rot =“);
  delay(100);
  Serial.println(a);
  Serial.print(„Blau =“);
  delay(100);
  Serial.println(b);
  Serial.print(„Gruen =“);
  delay(100);
  Serial.println(c);

randy = random(100,256);

  digitalWrite(Spot, HIGH);
  digitalWrite(Nebel, HIGH);

  farbe(0, 0, 0);

  a = analogRead(light);  //Lesen des aktuellen
Wertes des InPins und speichern in der Variable.
  b = analogRead(light1);
  c = analogRead(light2);

  if (a > 550) {

    a = analogRead(light);
    b = analogRead(light1);
    c = analogRead(light2);
    delay(500);
    digitalWrite(Nebel, LOW);
    delay(1000);

   digitalWrite(Spot, LOW);
    delay(1500);
    farbe (map(b, _from, _to, 0, 255), randy ,map(c,
_from, _to, 0, 255));  //
    delay(15000);

  }
  else {

  }

  digitalWrite(Nebel, HIGH);
  delay(2000);

  farbe(0, 0, 0);
  digitalWrite(Spot, HIGH);




}
void farbe(int r, int g, int b) {

  for (int i = 0; i <= N_LEDS; i++) {
    strip.setPixelColor(i, r, g, b);
  }
  strip.show();

}

//
```

Je mehr Lichtkraft der Mensch durch spirituelle Lebensführung und Hingabe an hohe Ideale entwickelt, desto schöner werden die Farben seiner Aura sein.

... konzentriere dich auf dich selbst – lass die Dinge, die außen sind hinter dir – finde deine innere Mitte – lass den Strömen freien Lauf ...

Dieser Schrein steht still in ausbalancierter Räumlichkeit. Es ist ein Ort der Ruhe und geschützt vor äußerer Strahlung. Nur ein Licht erreicht diesen heiligen Stein. Die Suchenden finden eine Fläche an der sie ihre Hände auslesen lassen können.

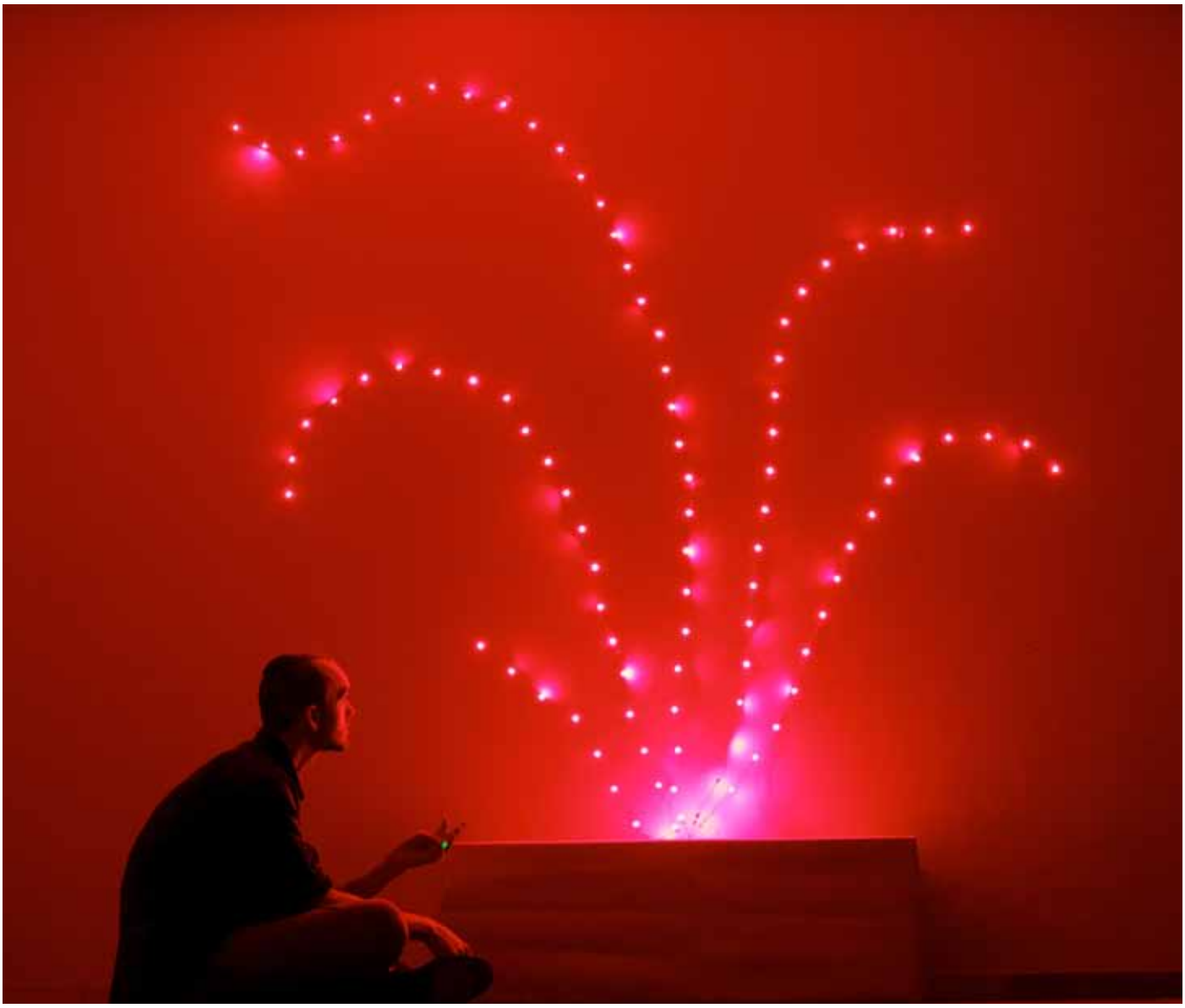
Es wird einige Sekunden dauern bis diese Fläche deine Spiritualität aufgenommen hat. Sobald das Licht ausgeht, siehst du deine innere Farbe. Unterstützend dazu wirst du den Klang der Seeligkeit vernehmen.



Bei genauerer Betrachtung erkennt man, dass es eine Grundfarbe der Aura gibt, welche in etwa der charakterlichen Beschaffenheit des Menschen entspricht. Darauf aufgelagert sind die Farben der momentanen Stimmung oder Emotionalität. Letztere Farben sind in andauernder Bewegung- Strömungen, Turbulenzen, Zunahme und Abnahme der Intensität, Wellen und Vibrationen.

Jede Farbe hat ihre entwickelten (leuchtende) und ihre unentwickelten (dunkle, matte) Tönungen: Klare, kräftige, lichtvolle Töne in der Aura zeigen positive Qualitäten an, wie Eifer, Kraft und Willensstärke. Matte, dunkle und dumpfe Töne zeigen Mangel an Kraft und Stabilität.

https://www.wirkendekraft.at



Heartbeat

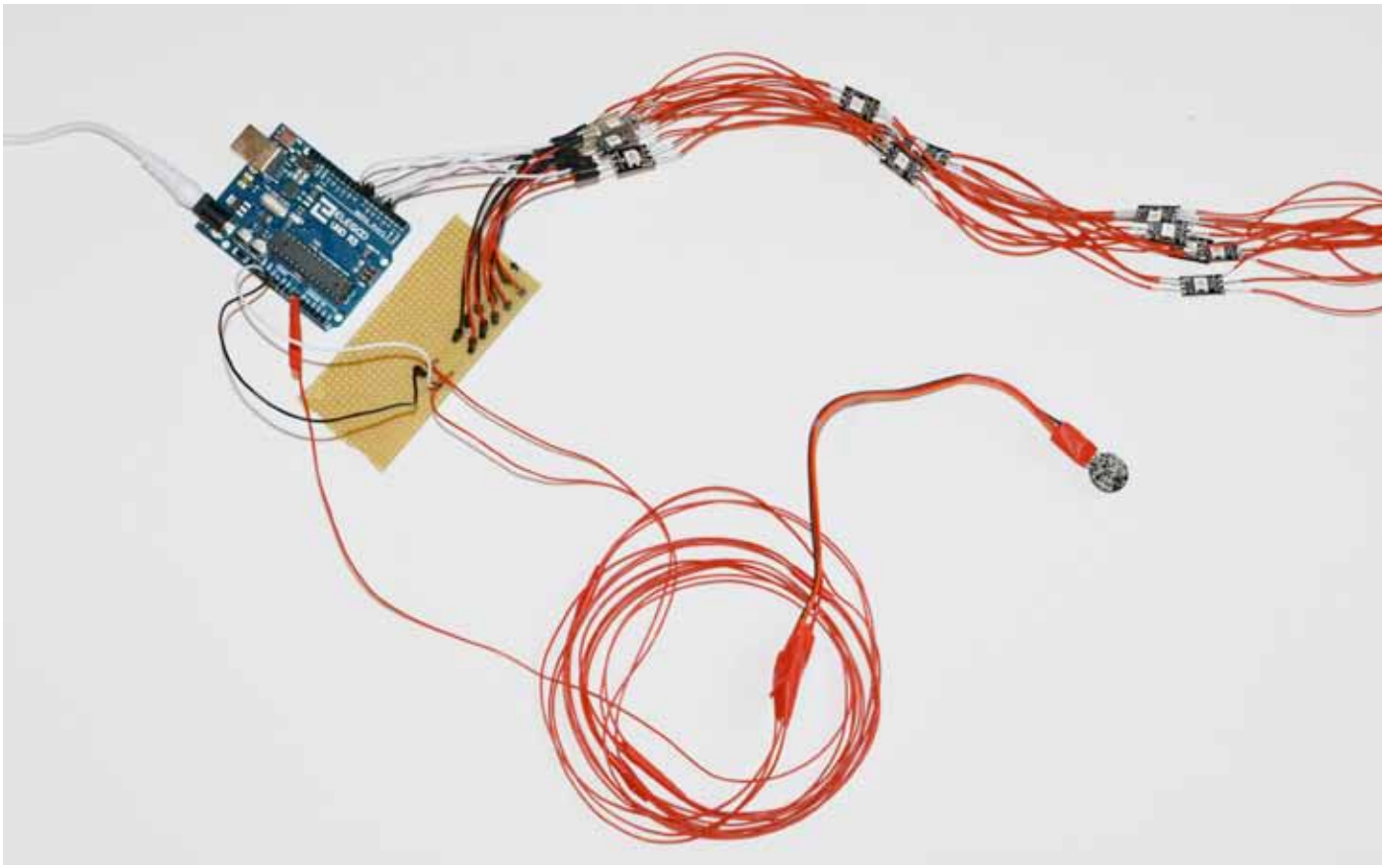
Experiment von Milan Behrens

Programmcode

```
#include <Adafruit_NeoPixel.h>

#define N_LEDS 30
#define PIN    3
#define PIN22  4
#define PIN33  5
#define PIN44  6
#define PIN55  7


int PulseSensor = 0;
int Signal;
int Treshold = 530;

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, PIN, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel strip2 = Adafruit_NeoPixel(N_LEDS, PIN22, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel strip3 = Adafruit_NeoPixel(N_LEDS, PIN33, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel strip4 = Adafruit_NeoPixel(N_LEDS, PIN44, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel strip5 = Adafruit_NeoPixel(N_LEDS, PIN55, NEO_GRB +
NEO_KHZ800);


void setup() {
 strip.begin();
 strip2.begin();
 strip3.begin();
 strip4.begin();
 strip5.begin();

 Serial.begin(9600);

}

void loop() {

Signal = analogRead(PulseSensor);
if(Signal > Treshold){

  setRed();

} else {
  setBlack();
  }
}



void setBlack(){
   for(int i = 0; i<=N_LEDS; i++){
    strip.setPixelColor(i, 0,0,0);
    strip2.setPixelColor(i, 0,0,0);
    strip3.setPixelColor(i, 0,0,0);
    strip4.setPixelColor(i, 0,0,0);
    strip5.setPixelColor(i, 0,0,0);
  }
  strip.show();
  strip2.show();
  strip3.show();
  strip4.show();
  strip5.show();
}

void setRed(){
   for(int i = 0; i<=N_LEDS; i++){
    strip.setPixelColor(i, 255,0,0);
    strip2.setPixelColor(i, 255,0,0);
    strip3.setPixelColor(i, 255,0,0);
    strip4.setPixelColor(i, 255,0,0);
    strip5.setPixelColor(i, 255,0,0);
  }
  strip.show();
  strip2.show();
  strip3.show();
  strip4.show();
  strip5.show();
}
```

Heartbeat

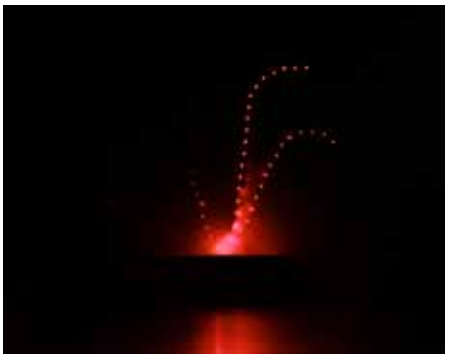
Experiment von Milan Behrens

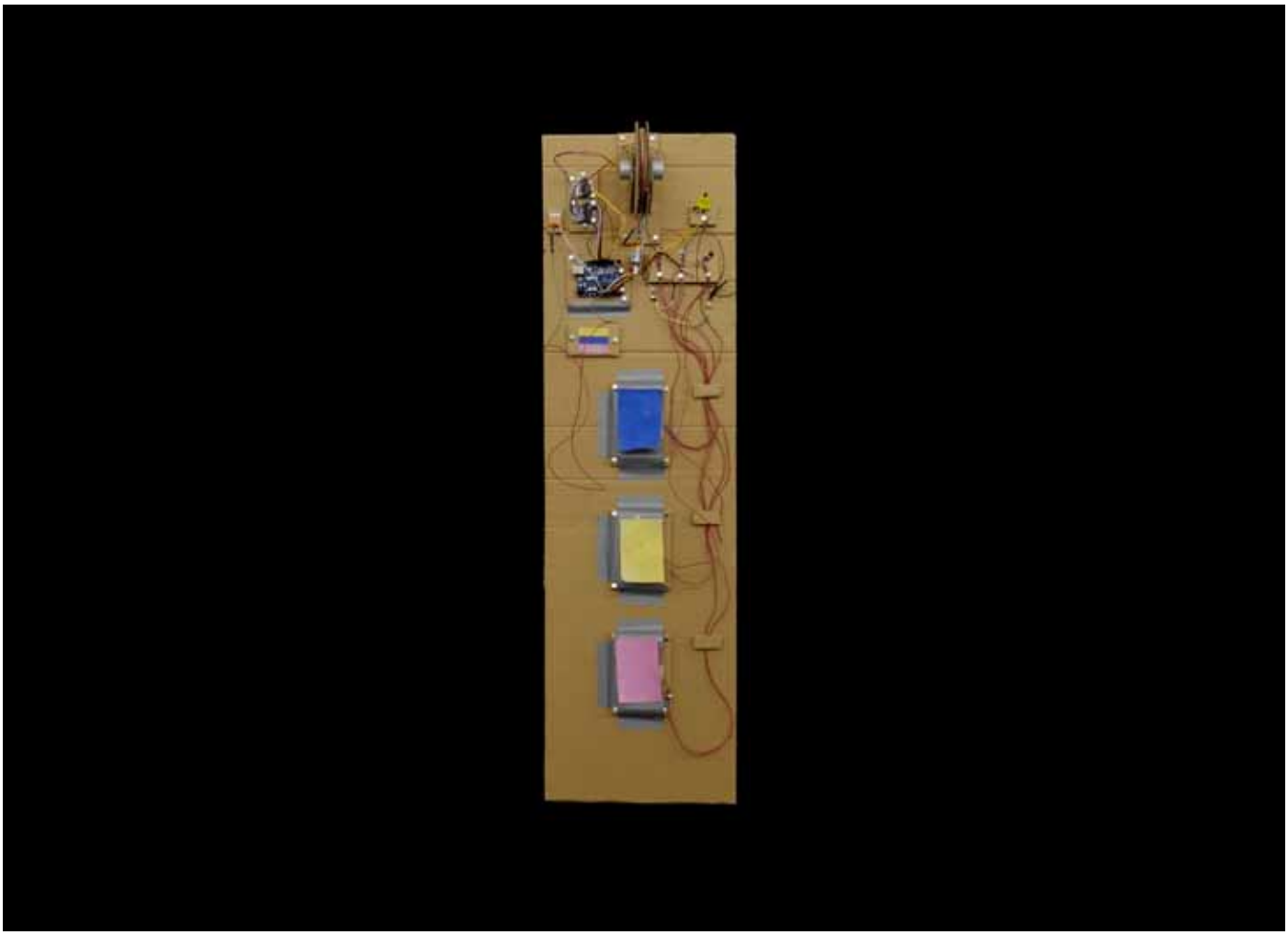


Da Eletronik, meinem Enpfinden nach, immer auch etwas kaltes anhaftet, fand ich es von Beginn an sehr reizvoll, sie mit etwas Lebendigem zu verbinden.

Und da das Herz das elementare Symbol für das Lebendige ist, habe ich mich dazu entschlossen meinen eigenen Puls als Input für mein Experiment zu nutzen.

Grundlegend für meinen Versuch war der Sensor. Man legt in an einer Stelle auf die Haut an der man seinen Puls ohnenhin stark spürt. Der Sensor sendet dann Lichtsignale aus welche die durchlaufende Menge an Blut messen.
Ich habe also diesen Sensor mit ca. neunzig einzeln verlötetetn RGB LED‘s verbunden und hab so eine live Übertragung meines Pulses erzeugt.





Ein kleines Glückspiel mit dem Arduino

Experiment von Nikolaus Hößle

Programmcode

```
#include <Stepper.h>
const int drehung = 2096;
Stepper myStepper1(200,6,7,8,9);
Stepper myStepper2(200,10,11,12,13);
#define fadePin 3
int Pin3 = 3;
int PiezoPin1 = A0;
int PiezoPin2 = A1;
int PiezoPin3 = A2;
int InPinValue1 = 0;
int InPinValue2 = 0;
int InPinValue3 = 0;
int farbe = 0;
int schwellenwert = 1000;
void setup() {
 pinMode(fadePin, OUTPUT);
  myStepper1.setSpeed(60);
  myStepper2.setSpeed(60);
   pinMode(PiezoPin1, INPUT);
  pinMode(PiezoPin2, INPUT);
  pinMode(PiezoPin3, INPUT);
Serial.begin(9600);
  randomSeed(analogRead(0));
 wuerfeln();
}
void loop() {
  Serial.println(farbe);
 InPinValue1 = analogRead(PiezoPin1);
  InPinValue2 = analogRead(PiezoPin2);
  InPinValue3 = analogRead(PiezoPin3);
 Serial.println(InPinValue1);
  Serial.println(InPinValue2);
  Serial.println(InPinValue3);
wuerfeln();
//->Teil1
  if(InPinValue1 >
schwellenwert){
     if(farbe == 1){
      for(int i = 0; i <=
drehung/2; i++){
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       freude();
       delay(1000);
       for(int i = 0; i <= drehung/2; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
       }
       }
    }
   if(InPinValue1 > schwellenwert){
   if(farbe == 2){
     for(int i = 0; i <= drehung/2; i++){
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       wut1();
       delay(1000);
       for(int i = 0; i <= drehung; i++){
         myStepper1.step(-1);
         myStepper2.step(1);
     }
     delay(1000);
     wut2();
      delay(1000);
     for(int i = 0; i <=
drehung*1.5; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
      }
      }
      }
   if(InPinValue1 >
schwellenwert)
{
   if(farbe == 3){
     for(int i = 0; i <=
drehung*0.5; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       wut2();
       delay(1000);
       for(int i = 0; i <=
drehung*0.5; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
     }
     delay(1000);
     wut1();
      delay(1000);
          for(int i = 0; i <=
drehung;
 i++){
         myStepper1.step(1);
         myStepper2.step(-1);
       }
    }
     }
  //->Teil2
  if(InPinValue2 >
schwellenwert)
{
     if(farbe == 2){
       for(int i = 0; i
<= drehung; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       freude();
       delay(1000);
       for(int i = 0; i
<= drehung; i++)
{
         myStepper1.step(1);
         myStepper2.step(-1);
       }
          }
          }
//-> hier die anderen Fälle
    if(InPinValue2
> schwellenwert)
{
    if(farbe == 1){
      for(int i = 0; i <=
drehung; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       wut1();
       delay(1000);
       for(int i = 0; i
<= drehung/2; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
     }
     delay(1000);
     wut2();
      delay(1000);
     for(int i = 0; i <=
drehung*1.5; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
      }
      }
      }
      }
   if(InPinValue2
> schwellenwert)
{
   if(farbe == 3){
     for(int i = 0; i
<= drehung*0.5; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       wut2();
       delay(1000);
       for(int i = 0; i
<= drehung*0.5; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
     }
     delay(1000);
     wut1();
      delay(1000);
          for(int i = 0; i <= drehung; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
       }
     }
     } //->Teil3

   if(InPinValue3
> schwellenwert)
{
     if(farbe == 3){
       for(int i = 0; i
<= drehung*1.5; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       freude();
       delay(1000);
       analogRead(PiezoPin3);
            for(int i = 0; i <= drehung*1.5; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
     }
        }
     }
 //-> hier die anderen Fälle
  if(InPinValue3
> schwellenwert)
{
      if(farbe == 1){
     Serial.println(„Fehler!“);
     for(int i = 0; i
<= drehung; i++)
{
         myStepper1.step(-1);
         myStepper2.step(1);
       }
       delay(1000);
       wut2();
       delay(1000);
       for(int i = 0; i <= drehung*0.5; i++){
         myStepper1.step(-1);
         myStepper2.step(1);
     }
     delay(1000);
     wut1();
      delay(1000);
         for(int i = 0; i <= drehung*1.5; i++){
         myStepper1.step(1);
         myStepper2.step(-1);
       }
```

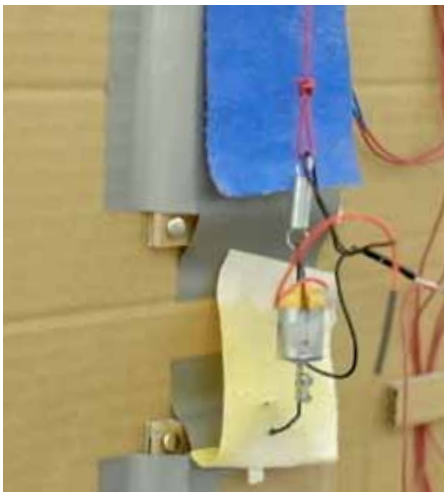
Ein kleines Glückspiel mit dem Arduino

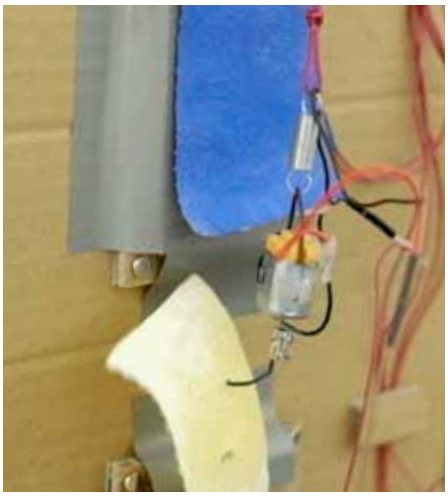
Experiment von Nikolaus Hößle

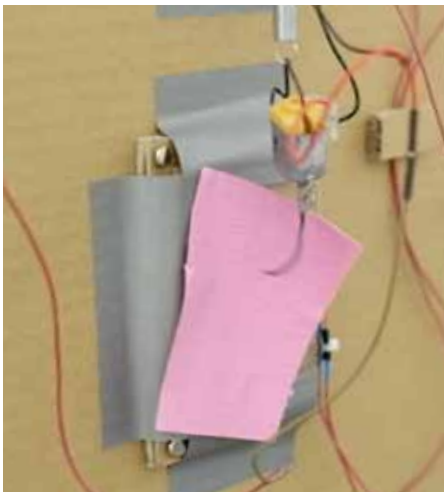
Welche Lieblingsfarbe haben unsere intelliegenten Toilettenpapierspender im 22. Jahrhundert ?

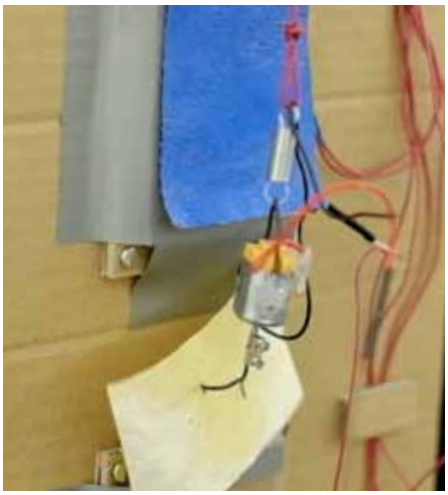
Es gibt 3 Farben zur Auswahl, eine davon hat sich Arduino als Lieblingsfarbe ausgewählt. Welche das ist, weiß man noch nicht. Jetzt wird ein Tipp abgegeben, welche Lieblingsfarbe Arduino haben könnte. Dies geschieht durch Berühren oder Antippen einer der 3 Farben.

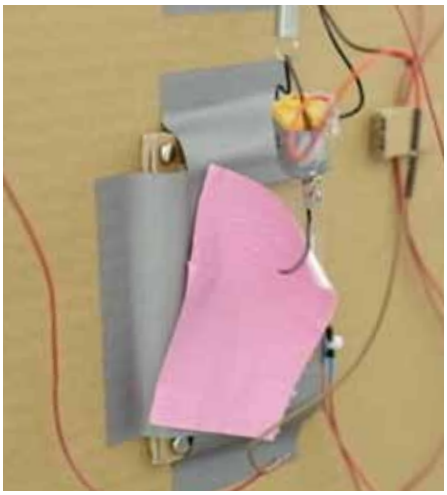
Arduino fährt nun mit Hilfe einer Seilwinde einen Motor auf die ausgewählte Farbe. Ist es seine Lieblingsfarbe, zuckt er ein paar Mal fröhlich. Ist es nicht seine Lieblingsfarbe, rotiert er wütend um sich selbst und reißt so die Farbe ab. Arduino sucht sich erst eine neue Lieblingsfarbe aus, wenn einmal die richtige Farbe getippt wurde.



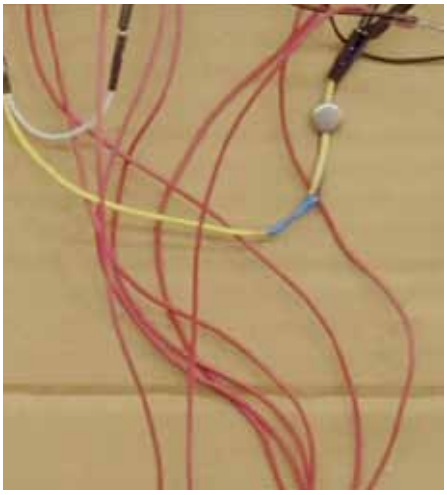




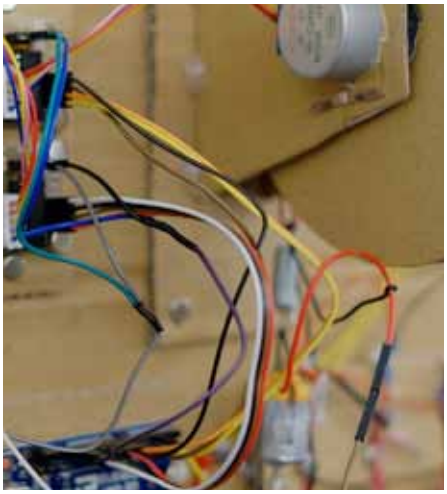




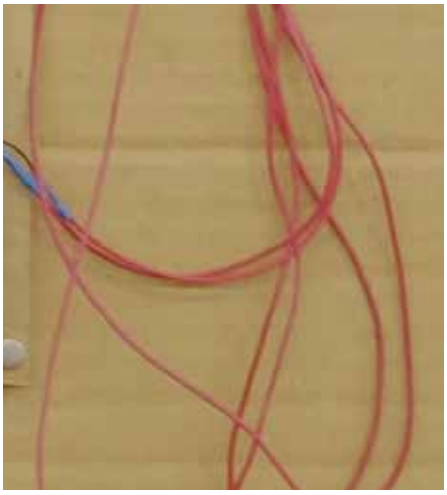
```
    }
    }
    if(InPinValue3
> schwellenwert)
{
        if(farbe == 2){
          Serial.println(„Fehler!");
      for(int i = 0; i <= drehung/2; i++){
          myStepper1.step(-1);
          myStepper2.step(1);
      }
      delay(1000);
      wut1();
      delay(1000);
      for(int i = 0; i
<= drehung; i++)
{
          myStepper1.step(-1);
          myStepper2.step(1);
    }
    delay(1000);
    wut2();
     delay(1000);

     for(int i = 0; i
<= drehung*1.5; i++)
{
          myStepper1.step(1);
          myStepper2.step(-1);
    }
    }
  }
} void wuerfeln(){
 farbe = random(1,4);

 }

 void freude(){
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(500);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

  delay(300);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(400);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

  delay(500);

  digitalWrite(Pin3,HIGH);
  delay(5);

 digitalWrite(Pin3,LOW);

  delay(300);


  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);
delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);

  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);


 }
 void wut1(){
  digitalWrite(Pin3,HIGH);
  delay(200);
 digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(50);

  digitalWrite(Pin3,HIGH);
  delay(100);
  digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(20);

 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(200);
  digitalWrite(Pin3,LOW);
  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(2000);
  digitalWrite(Pin3,LOW);


 }

 void wut2(){


digitalWrite(Pin3,HIGH);
  delay(100);
  digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(20);

 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(200);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

 delay(100);
  digitalWrite(Pin3,HIGH);
  delay(200);
 digitalWrite(Pin3,LOW);
```

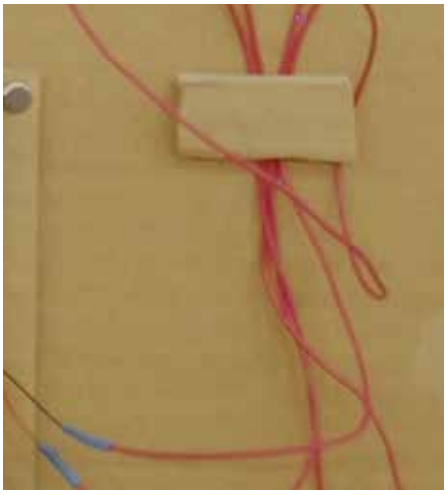


Mitte: 3 PiezoPins
Unten: Detailaufnahmen des Aufbaus



Unter den Farben habe ich PiezoPins so angeschlossen, dass sie Vibrationen wahrnehmen.
Ein PiezoPin besteht aus zwei Metallmembranen, die normalerweise so angesteuert werden, dass sie Klick- und Summgeräusche von sich geben. Durch die PiezoPins bekommt Arduino mitgeteilt, welche Farbe getippt wurde.





```
  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);
digitalWrite(Pin3,HIGH);
  delay(100);
  digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(20);
 igitalWrite(Pin3,HIGH);
  delay(100);
  digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(20);
igitalWrite(Pin3,HIGH);
  delay(100);
  digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(20);

 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(200);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

delay(100);
  digitalWrite(Pin3,HIGH);
  delay(200);
 digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(50);

  digitalWrite(Pin3,HIGH);
  delay(300);

 digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(500);
  digitalWrite(Pin3,LOW);

  delay(200);

  digitalWrite(Pin3,HIGH);
  delay(5);
  digitalWrite(Pin3,LOW);

  delay(100);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,HIGH);
  delay(10);
 digitalWrite(Pin3,LOW);

  delay(100);
  digitalWrite(Pin3,HIGH);
  delay(200);
 digitalWrite(Pin3,LOW);

  delay(10);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(50);

  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);

  delay(200);
  digitalWrite(Pin3,HIGH);
  delay(5);

digitalWrite(Pin3,HIGH);   digitalWrite(Pin3,HIGH);
  delay(5)digitalWrite(Pin3,HIGH);
  delay(5);

   digitalWrite(Pin3,HIGH);
  delay(3300);
  digitalWrite(Pin3,LOW); }


void chill(){
  digitalWrite(Pin3,HIGH);
  delay(10);
  digitalWrite(Pin3,LOW);
  delay(1000);
  }
```

Raumstation Partypause

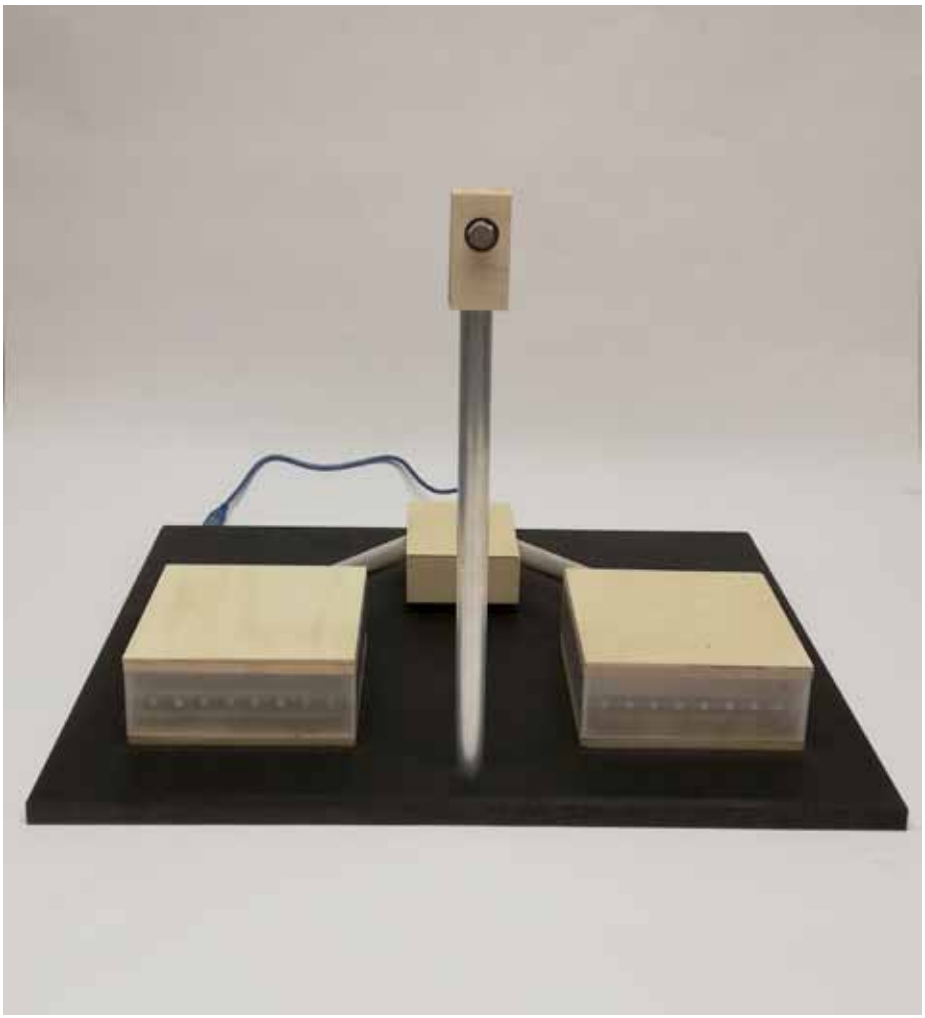
Experiment von Pierre Lichtenstein und Viola Nauck

Programmcode

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif

#define PIN 6

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic ‚v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(30, PIN, NEO_GRB + NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input
// and minimize distance between Arduino and first pixel.  Avoid connecting
// on a live circuit...if you must, connect GND first.

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
  #include <avr/power.h>
#endif

#define PIN 7

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic ‚v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)
Adafruit_NeoPixel stripl = Adafruit_NeoPixel(30, PIN, NEO_GRB + NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input
// and minimize distance between Arduino and first pixel.  Avoid connecting
// on a live circuit...if you must, connect GND first.

#define sensor A0

void setup() {
  // This is for Trinket 5V 16MHz, you can remove these three lines if you are not
using a Trinket
  #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif
  // End of trinket special code

 // This is for Trinket 5V 16MHz, you can remove these three lines if you are not
using a Trinket
  #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif
  // End of trinket special code

 Serial.begin(9600);
  delay(2000);
  pinMode(sensor, INPUT);

 stripl.begin();
  stripl.show(); // Initialize all pixels to ‚off'


  strip.begin();
  strip.show(); // Initialize all pixels to ‚off'
}

void loop() {
  float adcValue=0;
  for(int i=0;i<10;i++)
  {
    adcValue+= analogRead(sensor);
    delay(10);
  }
    float v= (adcValue/10) * (5.0/1024.0);
    float mgL= 0.67 * v;
    Serial.print(„BAC:");
    Serial.print(mgL);
    Serial.print(„ mg/L");
    if(mgL > 0.8)
    {

      Serial.println(„   Drunk");
      colorWipe(strip.Color(255, 0, 0), 50); // Red // Wein
      colorWipe(stripl.Color(0, 255, 0), 50); // Green// Wasser
    }
    else
    {

      Serial.println(„   Normal");
     colorWipe(strip.Color(0, 255, 0), 50); // Green //
     colorWipe(stripl.Color(255, 0, 0), 50); // Red // Wasser
    }
    delay(100);



}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
  // Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    stripl.setPixelColor(i, c);
    stripl.show();
    delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
  for(j=0; j<256; j++) {
    for(i=0; i<stripl.numPixels(); i++) {
      stripl.setPixelColor(i, Wheel((i+j) & 255));
    }
    stripl.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< stripl.numPixels(); i++) {
      stripl.setPixelColor(i, Wheel(((i * 256 / stripl.numPixels()) + j) & 255));
    }
    stripl.show();
    delay(wait);
  }
```

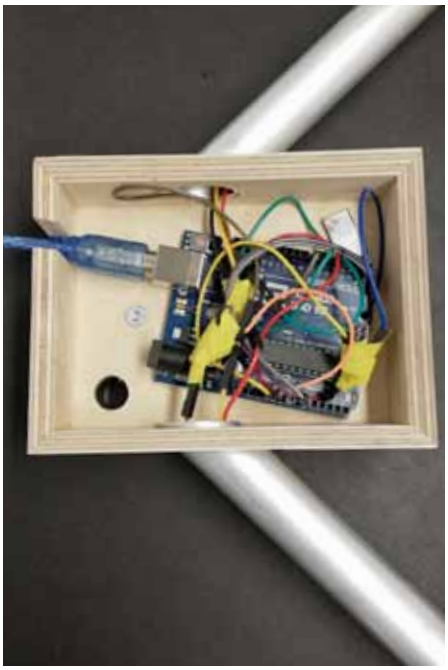
Raumstation Partypause

Experiment von Pierre Lichtenstein und Viola Nauck



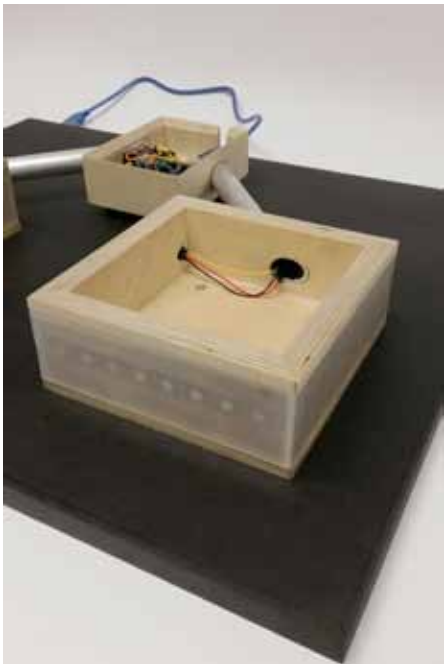
In dieser Kompaktwoche drehte sich alles um das Zusammenspiel aus einer Physical-Computing-Plattform und Farbe. Diese Soft- und Hardware wird unter dem Name Arduino zusammengefasst. Wir sollten uns nun eine Installation überlegen, welche diese beiden Komponenten-Technik und Farbe behandelt.

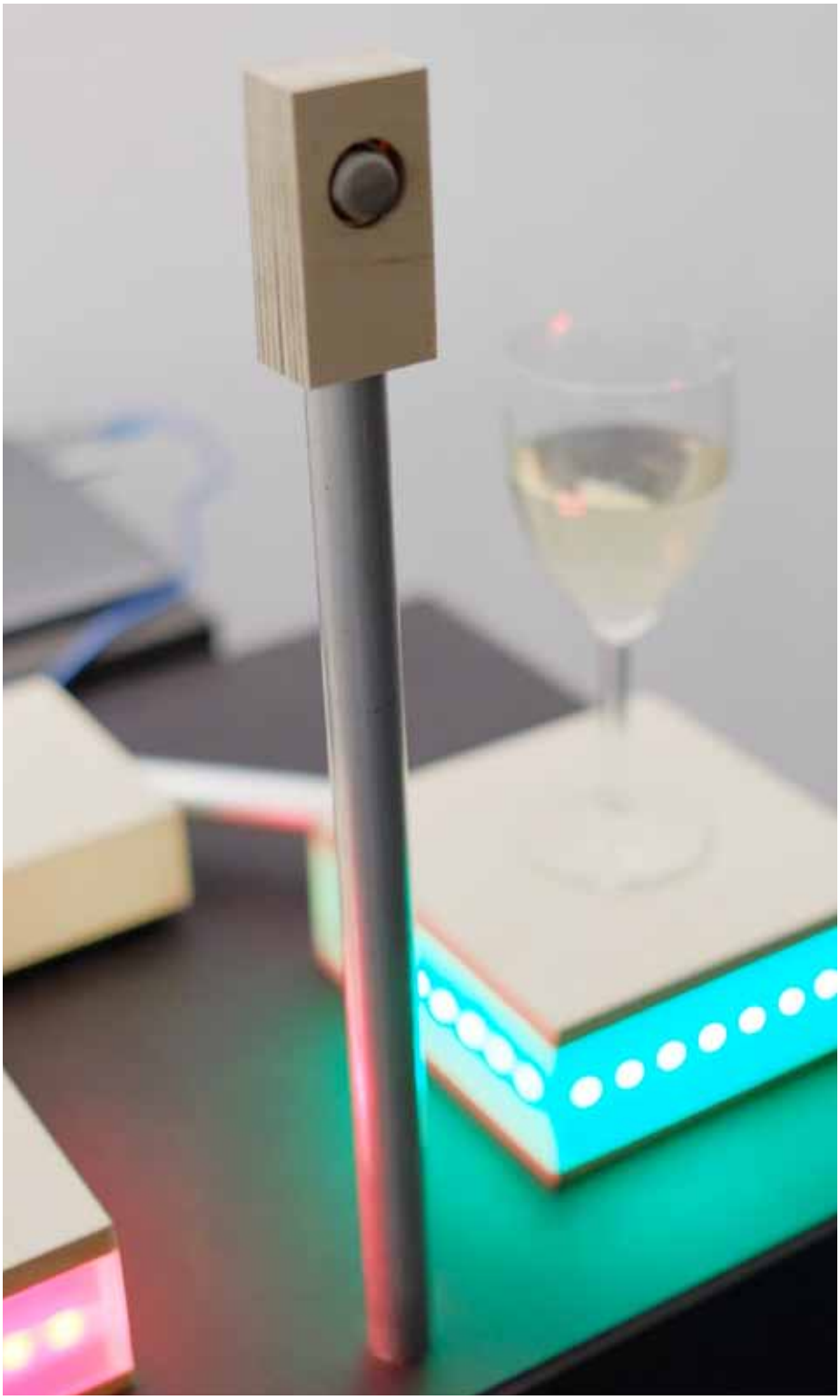
Es entstand ein Alkoholmessgerät, das dem Modell einer Raumstation ähnelt.



Auf einer scharzen MDF-Platte, befinden sich zwei größere und ein mittlerer Quader aus Multiplex. Diese sind über Aluminumrohre miteinander verbunden. Zudem gibt es noch ein weiteres Alurohr, welches mit einem noch kleinen Holzquader an der Spitze, in die Höhe ragt.

Im mittleren Quader sitzt das Herzstück, das Arduino Board. Dieses leitet die Lichtsignale an die LED's, die sich rund um die beiden großen Quader befinden. Welche Lichtsignale das sein sollen, bestimmt der Alkoholmesser, der im kleinen Quader sitzt.





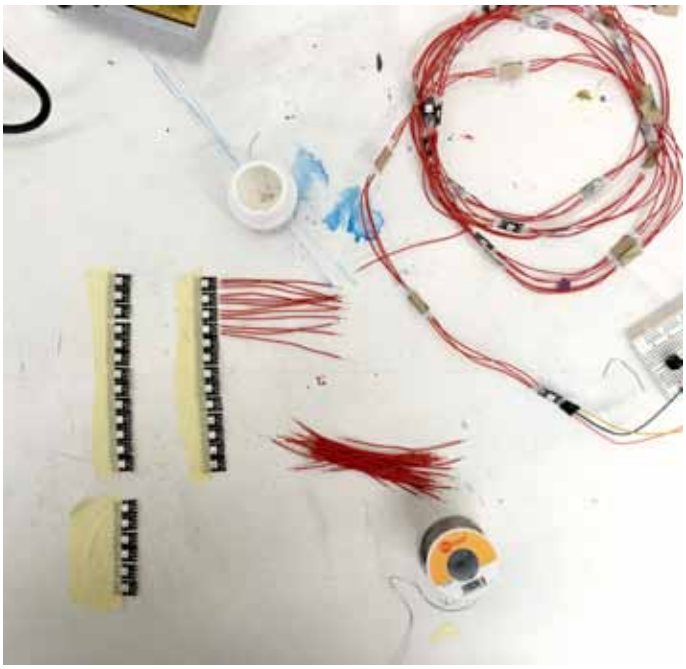
Bläst man in den Alkoholmesser, wird ein bestimmter Wert des aktuellen Alkoholanteils an das Arduino Board geschickt. Durch die von uns bestimmten Werte, leuchtet eine der beiden großen Quaderplattformen entweder rot oder grün. Auf den beiden Quadern befindet sich ein Wasserglas und ein Weinglas. Leuchtet die Plattform mit dem Weinglas grün, darf weiter angestoßen werden, leuchtet das Wasserglas grün, sollte der Alkoholkonsum pausiert werden.

RGB-Theremin
Ein durch Handbewegungen gesteuertes Farbinstrument

Experiment von Wayra Aguilar

Programmcode

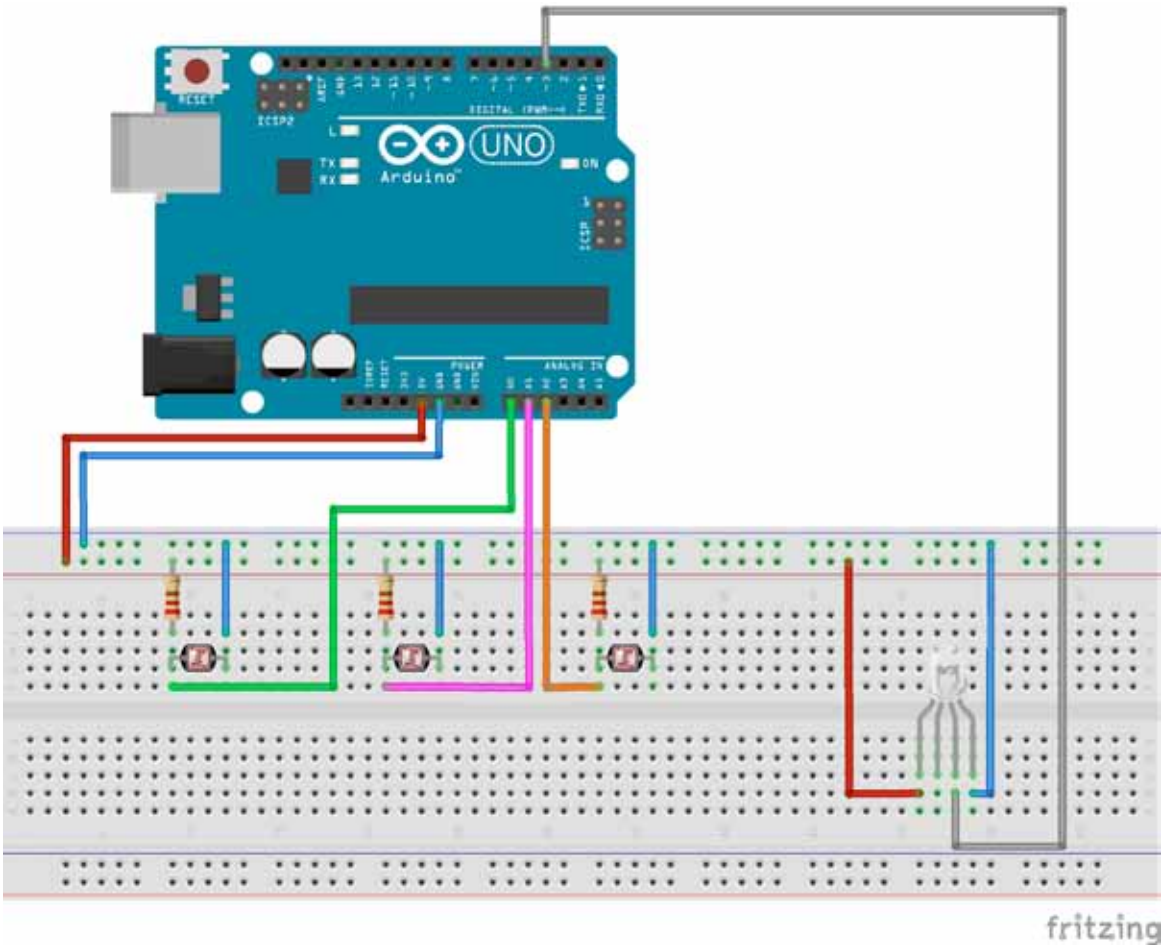
```
#include <Adafruit_NeoPixel.h>
  #ifdef __AVR__
  #include <avr/power.h>
  #endif
       int Pin1 = A0;
       int Pin2 = A1;
       int Pin3 = A2;

       int ldr1 = 0;
       int ldr2 = 0;
       int ldr3 = 0;

  #define PIN9  3
  #define PIN10  5
  #define PIN11  6

  #define NUMPIXELS     60

       Adafruit_NeoPixel pixels1 = Adafruit_NeoPixel(NUMPIXELS, PIN9, NEO_GRB + NEO_KHZ800);
       Adafruit_NeoPixel pixels2 = Adafruit_NeoPixel(NUMPIXELS, PIN10, NEO_GRB + NEO_KHZ800);
       Adafruit_NeoPixel pixels3 = Adafruit_NeoPixel(NUMPIXELS, PIN11, NEO_GRB + NEO_KHZ800);

void setup() {

  pinMode(Pin1, INPUT);
  pinMode(Pin2, INPUT);
  pinMode(Pin3, INPUT);
  Serial.begin(9600);

       #if defined (__AVR_ATtiny85__)
         if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
       #endif

  pixels1.begin();
  pixels2.begin();
  pixels3.begin();
}

void loop() {


  ldr1 = analogRead(Pin1);
  ldr2 = analogRead(Pin2);
  ldr3 = analogRead(Pin3);


       for(int i=0;i<NUMPIXELS;i++){

         pixels1.setPixelColor(i, pixels1.Color(ldr1/4,ldr2/4,ldr3/4));
         pixels2.setPixelColor(i, pixels2.Color(ldr1/4,ldr2/4,ldr3/4));
         pixels3.setPixelColor(i, pixels3.Color(ldr1/4,ldr2/4,ldr3/4));
         }
  pixels1.show();
  pixels2.show();
  pixels3.show();

}
```



Drei Photoresistoren dienen als Farbsteuerelemente. Je ein Resistor bestimmt den RGB-Fabrwert einer Farbe, also z.B. den Grünanteil von 0 – 255.

Die Resistoren reagieren auf Lichteinfall, d.h. man kann das darauffallende Licht mit der Handfläche langsam abdecken, je näher man kommt desto weniger Licht fällt auf den Sensor und somit sinkt der Wert der ausgelesen wird, dieser Wert wird dann in einen Rot-, Grün- oder Blauwert für die RGB LEDs übersetzt. Dadurch ist es möglich alle Farben im RGB Farbspektrum mit Handgestiken zu erzeugen.

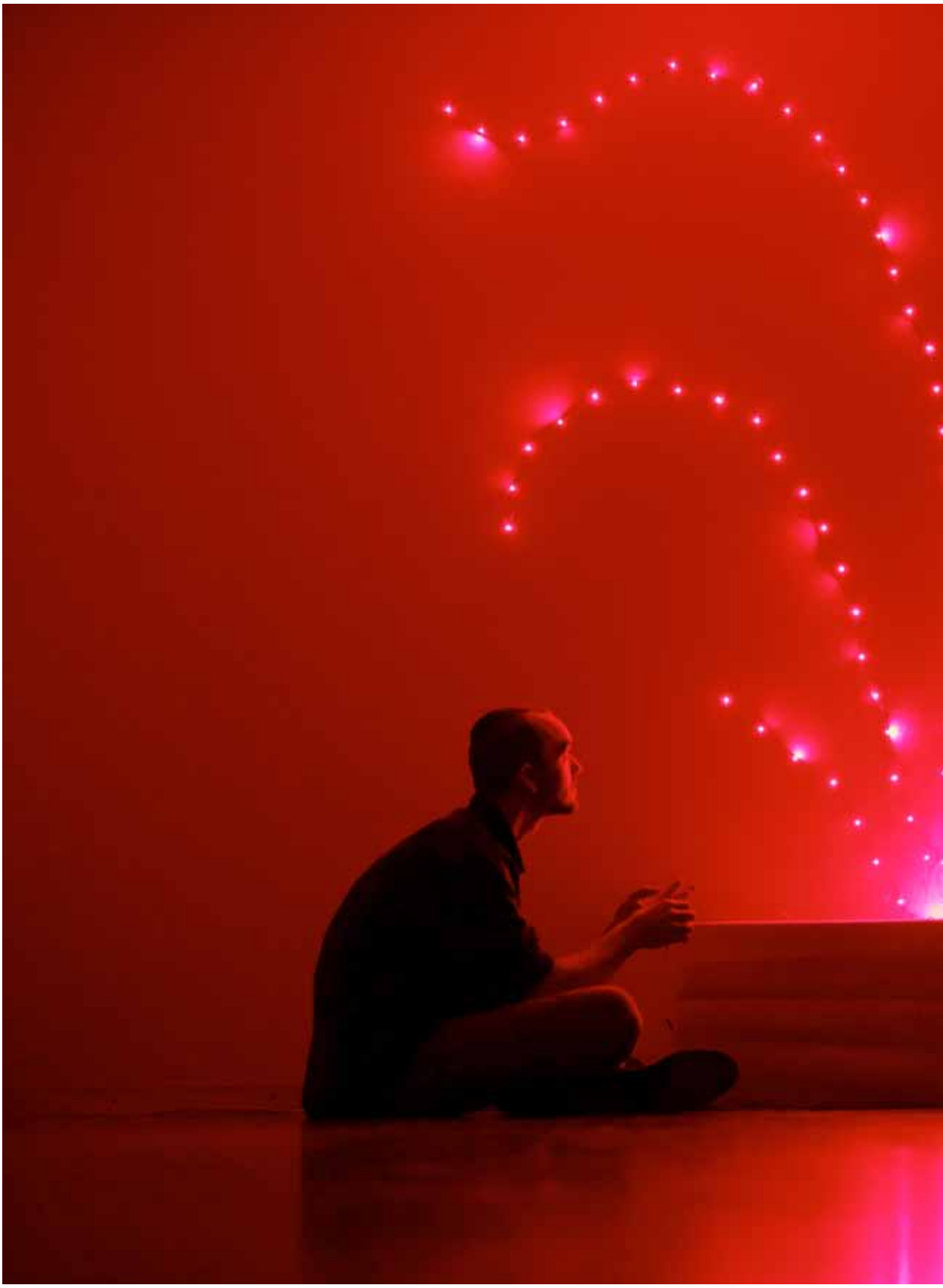
Dargestellt werden die Farben in diesem Versuchsaufbau mit Hilfe eines Leuchtkastens, der mit 145 LEDs ausgestattet ist und somit eine farbige Fläche, ähnlich eines Monitors, zeigt.

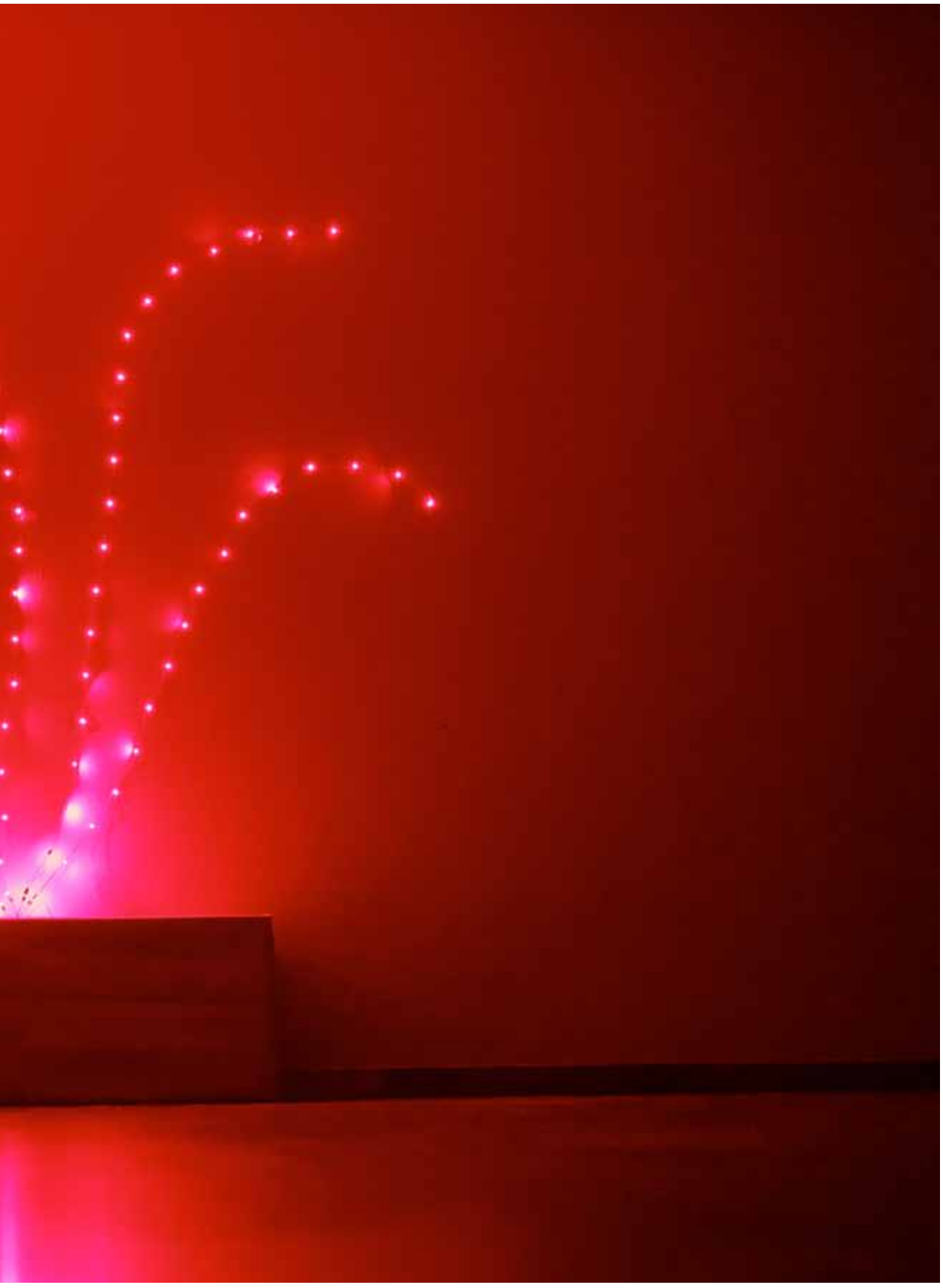


Der Leuchtkasten kann als eine Art Instrument verstanden werden, bei dem der Akteur hinter dem Objekt steht und für ein Publikum spielt, wie z.B. bei einem Theremin.

Für Zuschauer wird auf den ersten Blick nicht ersichtlich, wodurch die Farbe gemischt wird, da die Photoresistoren in der Oberfläche eingelassen sind. So wird Neugier ausgelöst, das Instrument zu erkunden und auszuprobieren, um zu verstehen wie es funktioniert.

4 Präsentation





id neuwerk
design education research

Burg Giebichenstein
Kunsthochschule Halle

2019